5-1-2013

A New Optimization Algorithm for Single Hidden Layer Feedforward Neural Networks

Leong Kwan Li
Hong Kong Polytechnic University

Sally Shao
Cleveland State University, s.shao@csuohio.edu

Ka-Fai Cedric Yiu
Hong Kong Polytechnic University

A new optimization algorithm for single hidden layer feedforward neural networks

Leong Kwan Li, Sally Shao, Ka-Fai Cedric Yiu

Introduction

Feedforward neural networks (FNNs) have been the subject of intensive research in recent years. FNNs are probably the most commonly used function approximation techniques in neural networks because of their capability of approximating any function with any desired accuracy providing some associated conditions being satisfied. Following the work of Hornik et al. [2], which showed that the Multi-layer feedforward networks are universal approximators in 1989, single hidden layer FNNs have been investigated more thoroughly out of many kinds of neural networks [4]. The classification and recognition properties of FNNs have been found in many papers such as Gibson [1] and Li [3]. It is clear that a single-l den layer FNN is sufficient to approximate the corresponding desired outputs arbitrarily close by the approximation theorem. FNNs have been applied to a wide variety of problems arising from a variety of disciplines, including mathematics, computer science, engineering, and medicine. However, the training/learning algorithm has a profound impact on the network learning capacity and its performance in modeling nonlinear dynamical phenomena [10,9]. Two issues are of great importance in the neural network training: how to avoid local minimum and how to achieve faster convergence. On the other hands, genetic algorithms (GAs) is a large class of learning algorithms based on the process of natural evolution. They are effective in function optimization with large and complex scale problems similar to neural networks. As a learning algorithm, genetic algorithms are promising but time consuming because GA might explore the global optimal solution with its stochastic property. Many researchers modify GA algorithms for their problem needs (see, for example [13,14,12,8]). Also, it is still a challenge in developing training algorithm that simultaneously improve convergence ability, learning speed, and generalization capability in real-world problems [5]. Viewing the error back propagation developed by Rumelhart et al. [11] is a piecewise gradient descent learning algorithm, Yam and Chow (1997) demonstrated an improved error back propagation algorithm to batch learning that one can use least squares technique that trains the FNN in a very efficient way. Later in Yam and Chow [16], they improved a weight initialization method for multi-layer FNN.

Classical GA techniques apply mutation and crossover on a string of one or more parents. This reproduction process is a discrete interchanging of parameters that may improve the population into next generation as discuss in Venkatesan et al. [15]. Kwong et al. [6] demonstrated how to use GA for training FNN. In particular, for a single-hidden layer FNN structure, we may generalize the idea of discrete learning process to a continuous one. In this paper, we present a new optimization algorithm which based on a particular learning strategy, namely the convex combination algorithm (CCA), to massaging the information in the hidden layer. Apart from the usual choice of the error function which sums the squares of the error norms at the output layer, the idea is to set up a new error function to measure the performance of the FNN, which the nonlinear optimization problem can be solved directly. Since the error function is a nonlinear function of the neural parameters, weights and thresholds, we then define an iterative process, the CCA learning algorithm to obtain the optimal choice of the connection weights. For a differentiable activation function σ, we may compute the gradient layer by layer systematically and minimize the error by gradient descent or quasi-Newton method. The CCA achieves the desired properties of convergence, fast learning speed, and easy generalization capability to tackle real-world problems. In the numerical experiments, our results shows that CCA has good exploration and exploitation capabilities in search of the optimal weight in training the single hidden layer FNN. We found that it is an effective training procedure comparing with some well-known least squares based algorithms as well as its descendants.

The organization of this paper is as follows. In Section 2, we present the methodology of optimization algorithm for the single hidden layer FNN. Let by this discussion, we show step by step how it can be improved to achieve better results and the desired learning properties in Section 2. In Section 3, we discuss the problem of evaluating and comparing the performance of the CCA in terms of numerical techniques. In the light of the discussion, we demonstrate some experimental results to show the effectiveness of the proposed optimization algorithm CCA. Finally, Section 4 is an account of concluding remarks.

Methodology

Suppose that we have a set of input and desired output pattern pairs (x_i, d_i), where the network inputs $x_i \in R^n$ and desired outputs $d_i \in R^m$ with $1 \le i \le N$. We use a single hidden layer FNN of n inputs and m outputs to learn the input and output relationship of the system. First, we assume that the number of hidden units, p, is larger than the number of output units m. That is, $p > m$. Let σ be the activation function which is bounded, differentiable and monotonic increasing. The network system output s_i for input x_i is defined by

$$s_i = Uy_i = U\sigma(Wx_i + \theta) \in R^m; \quad i = 1, 2, \ldots, n. \tag{1}$$

where the system outputs

$$y_i = \sigma(Wx_i + \theta),$$

$W = [w_{ij}]_{p\times n} \in R^{p\times n}$ and $U = [u_{ij}]_{m\times p} \in R^{m\times p}$ are the connection weight matrices; $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T \in R^n$ is the input bias or threshold vector of the system. With out loss of generality, we neglect the threshold θ in the following discussions by setting

$$x_{n+1} = 1, \quad w_{i(n+1)} = \theta_i.$$

That is, for simplicity, we have activation function for each layer except the outer layer which is "*identity*" function. Moreover, we use the identical activation function in all layers. Hence, for each $i = 1, 2, 3, \ldots, n$, we have $y_i = \sigma(\sum_{j=1}^{n+1} w_{ij}x_j) = \sigma(Wx_i) \in R^p$, which implies

$$s_i = Uy_i = U\sigma(Wx_i) \in R^m; \quad i = 1, 2, \ldots, n. \tag{2}$$

Consider the standard least squares error function defined by

$$\begin{aligned} E(U, W) &= \frac{1}{2}\sum_i^N \|s_i - d_i\|^2 \\ &= \frac{1}{2}\sum_i^N \|U\sigma(Wx_i) - d_i\|^2. \end{aligned} \tag{3}$$

The network is called *exactly capable* of the task if there exist some optimal connection weighted matrices U^* and W^* such that

$$E(U^*, W^*) = 0.$$

In practice, it is highly unlikely to obtain such U^* and W^* because of the numerical errors including the accumulative round-off errors. To due with this issue, the common practice is to provide some tolerances. We assume that there exist some U^* and W^* such that $E(U^*, W^*) < \epsilon$ for some prescribes tolerance ϵ. Hence, for a nonlinear least squares optimization problem, gradient or quasi-Newton type learning algorithms can be used to minimize the error function E. For instance, the error back propagation developed by Rumelhart et al. [11] is well known iterative learning algorithm. For a single-hidden layer FNN, since σ is differentiable, we may study the relation between the optimal weights at the minima. Along this line, we establish a new learning algorithm.

2.1. New error function

Let us first look at the optimization process of minimizing the error function. It is clear that the gradient of the error function ∇E has the form

$$\nabla E(U, W) = \begin{pmatrix} \dfrac{\partial E}{\partial U} \\ \\ \dfrac{\partial E}{\partial W} \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_i^N (s_i - d_i)y_i^t \\ \\ \displaystyle\sum_i^N U^t(s_i - d_i)y_i' x_i^t \end{pmatrix} \tag{4}$$

where g^t denotes the transpose of vector g. To minimize the error E, the necessary condition is

$$\nabla E(U^*, W^*) = 0.$$

It follows that $\partial E/\partial U = 0_{m\times p}$, which implies that the optimal U^* satisfies the equation

$$\frac{\partial E}{\partial U} = \sum_i (U^* y_i - d_i)y_i^t = 0 \tag{5}$$

and we obtain

$$U^* = \left[\sum_i d_i y_i^t\right]\left[\sum_i y_i y_i^t\right]^{-1}, \tag{6}$$

providing the inverse of $(\sum_i y_i y_i^t)$ exists. For given d_i, $i = 1, 2, \ldots, N$, let $Y_{p\times N} = [y_1, y_2, \ldots, y_N]$ and $T_{m\times N} = [d_1, d_2, \ldots, d_N]$ be the matrices of column vectors of outputs y_i and d_i, respectively. Then Eq. (6)

can be rewritten as

$$U^* = TY^t(YY^t)^{-1} = TY^+, \tag{7}$$

where $Y^+ = Y^t(YY^t)^{-1}$.

On the other hand, it is clear that the gradient $\partial E/\partial W$ involving the partial derivatives of the parameters U and W that cannot be solved directly. In view of this, instead of adding the momentum term or higher order terms to error functions as some researchers proposed, we propose in this paper to decouple the weight parameters U and W. The key idea is to define a new error function F to measure the errors at the hidden layer, that is,

$$\begin{aligned} F(U^+, W) &= \frac{1}{2}\sum_{i=1}^{N}\|U^+d_i - y_i\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}\|U^+d_i - \sigma(Wx_i)\|^2, \end{aligned} \tag{8}$$

where U^+ is the pseudo-inverse of U. Note that unlike the error function E in (3), the two connection weight matrices U^+ and W are now separated in the error function $F(U^+, W)$, which are fundamental important because it makes the computation of the gradients much easier and faster comparing with the standard error function E. In addition, it is not difficult to show that if the network is exactly capable of the task, that is, there exist some optimal connection weight matrices U^* and W^* such that $E(U^*, W^*) = 0$, we have $F(U^{+*}, W^*) = 0$.

To minimize the proposed error function $F(U^+, W)$, we look at the partial derivatives of the weight matrices U^+ and W in (8), which can be obtained in the similar manner as in (5),

$$\frac{\partial F}{\partial U^+} = \sum_{i=1}^{N}(U^+d_i - y_i)d_i^t.$$

Setting $\partial F/\partial U^+ = 0$, we obtain the optimal U^{+*}

$$U^{+*} = YT^t(TT^t)^{-1} = YT^+, \tag{9}$$

where

$$T^+ = T^t(TT^t)^{-1}. \tag{10}$$

We note that T^+ is a fixed matrix since $T = [d_1, d_2, \ldots, d_N]$ is a matrix of column vectors of d_i's, which are the given desired system outputs.

2.2. The learning algorithm at the hidden layer

To develop the new optimization algorithm for the given single hidden layer FNN, we consider least squares problem $F(U^+, W) = 0$. Given an arbitrary initial weights (U_0^+, W_0), we apply nonlinear least squares methods to compute the target values at the hidden layer. Suppose that we fix U_0^+ and optimize W_0 to W_1 for the error function F, which gives (U_0, W_1); then we fix W_1 and optimize U_0^+ to U_1^+ for the error function F to obtain (U_1^+, W_1); fixed U_1^+ and optimize W_1 to W_2 for the error function F, which gives (U_1, W_2). Repeat the processes, we obtain a sequence of connection weight matrix pairs in the following manner,

$$\begin{gathered} (U_0^+, W_0) \\ (U_0^+, W_1) \\ (U_1^+, W_1) \\ (U_1^+, W_2) \\ (U_2^+, W_2) \\ (U_2^+, W_3) \\ (U_3^+, W_3) \\ (U_3^+, W_4) \\ \vdots \\ \vdots \end{gathered} \tag{11}$$

with

$$F(U_0^+, W_0) \geq F(U_0^+, W_1) \geq F(U_1^+, W_1) \geq F(U_1^+, W_2) \geq \cdots > 0 \tag{12}$$

It can be summarized as the following sequence,

$$\{(U_{2n}^+, W_{2n}), (U_{2n}^+, W_{2n+1}), (U_{2n+1}^+, W_{2n+1}), (U_{2n+1}^+, W_{2n+2})\}_{n=0}^{\infty} \tag{13}$$

Rename the sequence as $\{(U_k^+, W_k)\}_k$, then $\{F(U_k^+, W_k)\}_k$ is a bounded monotonic decreasing sequence. Thus, we have the follow theorem.

Theorem 1. *The learning algorithm associated with the connection weight matrix pair sequence $\{(U_k^+, W_k)\}_k$ is convergent. Moreover, there exists a convergent subsequence $\{F_s(U_n^+, W_n)\}_{n=0}$ of $\{F(U_k^+, W_k)\}_k$ associated with the learning algorithm.*

Proof. First we need to show that the error function sequence $\{F(U_k^+, W_k)\}_k$ is convergent. By the construction which performs the optimization processes (12), the error function sequence $\{F(U_k^+, W_k)\}_{k=0}$ is bounded and monotonic decreasing. Thus, it converges to a minimum. By the Bolzano–Weierstrass theorem, $\{F(U_k^+, W_k)\}_{k=0}$ has a convergent sequence. We can construct a bounded and monotonic decreasing subsequence $\{F_s(U_n^+, W_n)\}_{n=0}$ from $\{F(U_k^+, W_k)\}_{k=0}$, which is also convergent to the minimum. □

Remarks. It is worth to point out that the sequence of the matrix pairs $\{(U_k^+, W_k)\}_k$ needs not to be convergent but the error function sequence $\{F(U_k^+, W_k)\}_{k=0}$ is convergent. We have a convergent learning algorithm associated with $\{F(U_n^+, W_n)\}_{n=0}$ for the single hidden layer FNN.

2.3. New optimization algorithm CCA

The learning algorithm that we obtained in Section 2.2 can be further improved. In this subsection, we construct another optimal learning algorithm CCA based on massaging the information in the hidden layer.

For the given input and desired output pattern pairs (x_i, d_i) and an arbitrary initial weights (U_0^{++}, W_0^+), let $Y_0 = \sigma(W_0^+X)$ be

the matrix of the column vectors y_i, $i = 1, 2, \ldots, N$, where X is the matrix of the column vectors x_i (the system inputs). For each i, we define $z_i = U^{++}d_i$ and Z_0 be the matrix of the column vectors of $U_0^{++}d_i$. Now Z_0 and Y_0 are expected information in hidden layer, where T is the matrix of the column vectors d_i as defined in (7). If Z_0 and Y_0 are equal, the error is zero and the FNN is exactly capable. Otherwise, we adjust the weight matrices to fit their convex combination $U_1^{++} = [\alpha Y_0 + (1-\alpha)Z_0]T^+$ for some $0 < \alpha < 1$, and minimize the error function F, where $T^+ = T^t(TT^t)^{-1}$ is as defined in (10). Thus, we update the matrix

$$U_1^{++} = [\alpha Y_0 + (1-\alpha)Z_0]T^+.$$

We assume that $\sigma^{-1}([\alpha Y_0 + (1-\alpha)Z_0])$ exists for each iteration, then we update W_0^+ by minimizing the error function F associated with the equation

$$W_1^{++}X = \sigma^{-1}([\alpha Y_0 + (1-\alpha)Z_0]),$$

where $X = [x_1, x_2, \ldots, x_N]$ is the matrix of column input vectors of x_i. Continue the processes, we obtain the following sequences of connection matrices $\{(U_k^{++}, W_k^+)\}_{k=1}$ iteratively,

$$Y_0 = \sigma(W_0^+X), \tag{14}$$

$$Z_0 = U_0^{++}T, \tag{15}$$

$$U_{k+1}^{++} = [\alpha Y_k + (1-\alpha)Z_k]T^+, \tag{16}$$

$$W_{k+1}^+X = \sigma^{-1}([\alpha Y_k + (1-\alpha)Z_k]).$$

In general, we may choose different α's in the above learning algorithm, say take $0 < \beta < 1$ in above equation, i.e.

$$W_{k+1}^+X = \sigma^{-1}([\beta Y_k + (1-\beta)Z_k]). \tag{17}$$

We called this optimization algorithm associated with $\{(U_k^{++}, W_k^+)\}_{k=1}$ the CCA. Since $T = [d_1, d_2, \ldots, d_N]$, which implies $T^+ = T^t(TT^t)^{-1}$ is a fixed matrix and it can be computed directly after the pattern pairs are given in batch learning. Hence, we can evaluate U_2^{++} from U_1^{++} and update U_k^{++} efficiently. No gradients need to be computed during the learning processes. Moreover, the CCA algorithm is convergent since the error functions $\{F(U_k^{++}, W_k^+)\}$ is a bounded monotonic decreasing sequence as it has been constructed.

We have the following convergence theorem of the learning algorithm CCA.

Theorem 2.

(i) *The CCA learning algorithm associated with the connection weight matrix pair sequence $\{F(U_k^{++}, W_k^+)\}_k$ is convergent.*
(ii) *The parameter α with $0 < \alpha < 1$, defined as in (16), can be adjusted to speed up the rate of convergence of the optimization algorithm CCA.*

Proof. By our construction of CCA, where

$$F(U_0^{++}, W_0^+) \geq F(U_1^{++}, W_1^+) \geq F(U_2^{++}, W_2^+) \geq \cdots > 0 \tag{18}$$

(i) follows.
(ii) Let $\lim_{n\to 0} F(U_n^{++}, W_n^+) = F(U_*^{++}, W_*^+)$. By the definition of F, (14)–(16), we have

$$\begin{aligned}
F(U_k^{++}, W_k^+) - F(U_*^{++} - W_*^+) &= \frac{1}{2}\left[\sum_{i=1}^{N} \|U_k^{++}d_i - \sigma(W_k^+X)\|^2 - \|U_*^{++}d_i - \sigma(W_*^+X)\|^2\right] \\
&= \frac{1}{2}\left[\sum_{i=1}^{N} (\|U_k^{++}d_i + \sigma(W_k^+X)\|)(\|U_*^{++}d_i - \sigma(W_*^+X)\|)\right] \\
&\geq [\|(U_k^{++} - U_*^{++})T - (\sigma(W_k^+X) - \sigma(W_*^+X)) + (U_*^{++}T - \sigma(W_*^+X)\|) + \|U_*^{++}T - \sigma(W_*^+X)\|]^2 \\
&\geq \left|\frac{1}{2}[\|(U_k^{++} - U_*^{++})T - (\sigma(W_k^+X) - \sigma(W_*^+X))\| - 2\|(U_*^{++}T - \sigma(W_*^+X)\|)]^2\right| \\
&= \left|\left(\frac{1}{2}\right)[\alpha\|(Z_k - Z^*) - (Y_k - Y^*)\| - 2\|Z^* - \alpha Y^* - (1-\alpha)Z^*\|)]^2\right| \\
&= \left(\frac{1}{2}\right)[\alpha|(\|(Z_k - Z^*) - (Y_k - Y^*)\| - 2\|Y^* - Z^*\|)]^2|,
\end{aligned} \tag{19}$$

where $Z^* = U_*^{++}T$, $Y^* = \sigma(W_*^+X)$, and $0 < \alpha < 1$. Note that as $k \to \infty$,

$$F(U_k^{++}, W_k^+) - F(U_*^{++}, W_*^+) \to 0,$$

which implies $Y^* \to Z^*$, the desired output matrix. Moreover, the sequence will converge faster when $\alpha \to 0^+$. We finish the proof. □

Remarks.

(i) There is an interesting observation about the sequence $\{U_k^{++}\}$. Consider

$$
\begin{aligned}
U_{k+1}^{++} - U_k^{++} &= [\alpha Y_k + (1-\alpha)Z_k - \alpha Y_{k_1} + (1-\alpha)Z_{k-1}]T^+ \\
&= [\alpha(Y_k - Y_{k_1}) + (1-\alpha)(Z_k - Z_{k-1})]T^+ \\
&= [\alpha(Y_k - Y_{k_1})T^+ + (1-\alpha)(U_k^{++} - U_{k-1}^{++})]TT^+ \\
&= [\alpha(Y_k - Y_{k_1})T^+ + (1-\alpha)(U_k^{++} - U_{k-1}^{++})] \\
&= [\alpha(Y_k - Y_{k_1})T^+ + (1-\alpha)([\alpha(Y_{k-1} - Y_{k-2}) + (1-\alpha)(Z_{k-1} - Z_{k-2})]T^+)] \\
&= [\alpha(Y_k - Y_{k-1}) + \alpha(1-\alpha)(Y_{k-1} - Y_{k-2})]T^+ + (1-\alpha)^2(U_{k-1}^{++} - U_{k-2}^{++}) \\
&\vdots \\
&\vdots \\
&= \alpha[(Y_k - Y_{k-1}) + \alpha(1-\alpha)(Y_{k-1} - Y_{k-2}) + \cdots + (1-\alpha)^{k-1}(Y_1 - Y_0)]T^+ + (1-\alpha)^k(U_1^{++} - U_0^{++}).
\end{aligned}
\tag{20}
$$

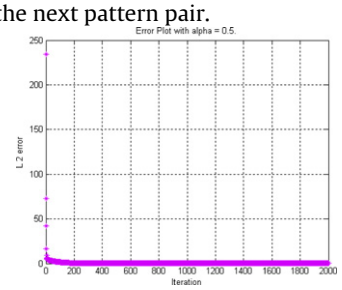
We notice that $\{U_k^{++}\}$ converges if and only if $\{Y_k = \sigma(W_k^+ X)\}$ converges.

(ii) Yam and Chow (1997) provided an analysis of the convergence rate for an extended least squares based algorithm for training multi-layer FNNs. Their result showed that the learning rat is depending on the correlation between the negative error gradient and the previous weight update of that layer. The analog analysis can be applied to our optimization algorithm CCA by replacing the the negative error gradient term to ratio of $\|Y_{k+1} - Y^*\|/\|Y_k - Y^*\|^{\lambda}$ for some $\lambda > 0$.

Numerical experiment

We have tested the above learning algorithm based on the least squares problem. The first numerical experiment is as follows: given the monthly data sunspot series data from 1900 to 2009, we normalize the values of the series as $s(k)$, $1 \le k \le 1320$, between -1 and 1. Then we use twelve successive subsequence data, $s(k)$, $s(k+1)$, ..., $s(k+11)$ as input data to approximate the coming 6 month data $s(k+13)$, $s(k+14)$, ..., $s(k+18)$.

We generate the input and output pair by shift 10 months data, i.e. $s(k+11)$, $s(k+7)$, ..., $s(k+24)$ to approximate $s(k+25)$, $s(k+26)$, ..., $s(k+30)$ as the next pattern pair.



We partition the sequence $s(k)$ into 100 data patterns for simplicity. A FNN with 12 input units, 12 hidden units and 6 linear output is used to approximate these patterns. The initial state of the neural parameters (U_0, W_0) are randomly generated integers between -5 and 5, and then updated as the learning algorithm described above. The initial error is 234.335. We tested 3 typical $\alpha = 0.382$, 0.5 and 0.618 In all cases, the errors $F(U^+, W)$ converge to 2.6107×10^{-11}, 7.4758×10^{-10} and 3.6877×10^{-8} respectively in 2000 iterations. All numerical experiments used Matlab program and finished within a few seconds. Since the error converges in a similar pattern, we just show one error plot for $\alpha = 0.5$.

In fact, deterministic learning algorithm is much more efficient than GA in practice. It will be good to have hybrid algorithm in optimization that improves on the efficiency as well as avoiding local minima. As our method converges much faster than those classical learning algorithms based on the gradient descent methods, it may be good to use GA for new initial starting point search and then employ our proposed method that converges in a few hundred iterations. If the solution is not within the prescribed tolerance, then we use GA again to move on another initial starting point till satisfactory.

The second example came from the soldering process in manufacture process. We took the handy normalized data in Table III of Kwong et al. [7]. We use a FFN with 4 inputs that represents the pressure, needle inner diameter, short size and dwell time, respectively. The inputs consist of two levels only, and we modify the inputs as ± 1 in our experiment instead of 0 and 1 in the original table. The hidden layer consists of 6 units. The FFN has 3 linear output units that represent the fuzzy solder spot diameters. Thus, the system outputs try to capture the mean, the left and right variations of the machine in soldering. There are 16 desired data.

We assume $\alpha = 0.75$ and $\beta = 0.7$. Also, we randomly choose the connection weights and thresholds between ± 2 and the initial error norm is 9.2445. Our PC took a few seconds to finish 10,000 iterations. The error norm was down to 4.0892×10^{-15}. As the error converges in the same manner as the above experiment, we omit the error plot here.

Concluding remarks

A new optimization algorithm CCA has been proposed that may be viewed as a combination of a GA algorithm and a modified back propagation learning. The numerical experiments demonstrated some impressive results that the CCA has good exploration and exploitation capabilities in searching for the optimal weight in training the single hidden layer FNN. This algorithm has the advantage that no computation of the derivative is required because we have decoupled the weight matrices. On the other hand, it achieves the desired properties of convergence, high learning speed, and easy generalization capability to tackle real-world problems. For future research, we may also consider other error function such as

$$G(U^+, W) = \frac{1}{2}\sum_{i=1}^{N} \|\sigma^{-1}(U^+ d_i) - W x_i\|^2$$

that decouple the weight matrices. In fact, the computation efficiency of this algorithm is enhanced because after we obtain $T^+ = T^t(TT^t)^{-1}$ in (9), T^+ can be used repeatedly in all iterations. Furthermore, the optimization algorithm CCA can be extended to FNN having two or more hidden layers. Nevertheless, it is an interesting problem and will need a lot of efforts in applying GA to this particular structure FNN. Our proposed method converges in a few hundred iterations but might be trapped by local minima. It is good to integrate with GA to search for a good initial starting point, then use our method to find a minimum as a hybrid learning method [17].

References

[1] G.J. Gibson, Exact classification with two-layer neural nets, Journal of Computer and System Sciences 52 (1996) 349–356.

[2] K. Hornik, M. Stinchcombe, H. White, Multi-layer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[3] L.K. Li, On computing decision regions with neural nets, Journal of Computer and System Sciences 43 (1991) 509–512.

[4] G.B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 1420–1434.

[5] D.A. Karras, S.J. Perantonis, An effective constrainted training algorithm for feedforward networks, IEEE Transactions on Neural Networks 6 (6) (1995) 879–892.

[6] C.K. Kwong, Y.K. Chan, M.E. Aydin, T.C. Fogarty, An orthogonal array based genetic algorithm for developing neural network for fluid dispensing process, International Journal of Production Research 44 (12) (2006) 4815–4836.

[7] C.K. Kwong, Y. Chen, K.Y. Chan, H. Wong, Hybrid fuzzy least-squares regression approach to modeling manufacturing processes, IEEE Transactions on Fuzzy Systems 16 (3) (2008) 644–651.

[8] G.P. Nicolas, O.B. Domingo, H.M. Cesar, An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization, Neural Networks 19 (4) (2006) 514–528.

[9] S. Razavi, B. Tolson, A new formulation for feedforward neural networks, IEEE Transactions on Neural Networks 22 (10) (2011) 1588–1598.

[10] J.J. Rubio, P. Angelov, J. Pacheco, Uniformly stable backpropagation algorithm to train a fedforward neural network, IEEE Transactions on Neural Networks 22 (3) (2011) 356–366.

[11] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representation by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), in: Parallel Distributed Processing, vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.

[12] C.Y. Shen, L.X. Wang, Q. Li, Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method, Journal of Materials Processing Technology (2007) 183, p412V418.

[13] C. Sivapathasekaran, S. Mukherjee, A. Ray, A. Gupta, R. Sen, Artificial neural network modeling and genetic algorithm based medium optimization for the improved production of marine biosurfactant, Bioresource Technology 101 (8) (2010) 2884–2887.

[14] J.T. Tsai, J.H. Chou, T.K. Liu, Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm, IEEE Transactions on Neural Networks 17 (1) (2006) 69–80.

[15] D. Venkatesan, A.K. Kannan, A.R. Saravanan, A genetic algorithm-based artificial neural network model for the optimization of machining processes, Neural Computing & Applications 18 (2009) 135–140.

[16] J.Y.F. Yam, T.W.S. Chow, A weight initialization method for improving training speed in feed forward neural network, Neurocomputing 30 (1–4) (2000) 219–232.

[17] K.F.C. Yiu, Y. Liu, K.L. Teo, A hybrid descent method for global optimization, Journal of Global Optimization 28 (2) (2004) 229–238.