Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Convergence of BP algorithm for product unit neural networks with exponential weights

C. Zhang, W. Wu*,1, X.H. Chen, Y. Xiong

Department of Applied Mathematics, Dalian University of Technology, Dalian 116024, PR China

Received 28 August 2007; received in revised form 30 November 2007; accepted 4 December 2007
Communicated by Z. Wang

Abstract

Product unit neural networks with exponential weights (PUNNs) can provide more powerful internal representation capability than traditional feed-forward neural networks. In this paper, a convergence result of the back-propagation (BP) algorithm for training PUNNs is presented. The monotonicity of the error function in the training iteration process is also guaranteed. A numerical example is given to support the theoretical findings.
© 2008 Elsevier B.V. All rights reserved.

MSC: 92B20; 68Q32; 74P05

Keywords: Neural network; Product unit; Exponential weights; Back-propagation algorithm; Convergence

1. Introduction

Traditional feed-forward neural networks are constructed by using multiple layers of summation units. These networks can effectively solve approximation and classification problems [3,9]. However, usually it requires a large number of summation units for a traditional feed-forward neural network to approximate a complicated function. Various kinds of "high order" neural networks have been developed to overcome this drawback [6,7,4]. Among them, product unit neural networks with exponential weights (PUNNs) were proposed by Durbin and Rumelhart [4], and studied in [12,11,10,19]. PUNNs are used to solve regression problems in [16,17]. But we have not found any theoretical analysis on the convergence of the training methods for PUNNs, and this becomes our main concern in this paper.

Back-propagation (BP) algorithm is possibly the most popular optimization algorithm for training feed-forward neural networks [21,22]. Unfortunately, the solution space for PUNNs can be extremely convoluted, with numerous local minima that trap the BP training (cf. [4,12,5]). Hence, some global optimization algorithms such as Genetic Algorithm [11], Random Search [12], Evolutionary Algorithms [14,15,8], Particle Swarm Optimization and Leap Frog Optimization [10], etc. have been used to train PUNNs. But these global optimization algorithms usually converge very slowly. So it seems natural that BP algorithm, which is good at local optimization, has been used to combine with some global search algorithms such as the Random Search Algorithm (RSA), resulting in quite satisfactory training [12]. In this strategy, BP is used to move to the local minima, and if the training error is still above the desired level, the RSA algorithm generates a new set of random weights from which BP can start again. So the convergence analysis of the BP algorithm for PUNN is still useful as a theoretical support of the above training strategy.

The organization of the rest of the paper is as follows. Section 2 introduces PUNN. Section 3 describes the learning procedures of BP for PUNN. The main convergence results are presented in Section 4. Experimental results are given in Section 5. Some conclusions are drawn

*Corresponding author.
E-mail address: wuweiw@dlut.edu.cn (W. Wu).
[1]Partly supported by the National Natural Science Foundation of China (10471017).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.12.004

in Section 6. Appendix A is an appendix, in which the details of the proofs are provided.

2. Product unit neural networks with exponential weights

In PUNN, the traditional summation units of the hidden layer are replaced by the product units with exponential weights:

$$\sum_{n=1}^{N} w_n x_n \quad \to \quad \prod_{n=1}^{N} x_n^{w_n}.$$

Due to the special exponential form, PUNNs have some additional advantages. Durbin and Rumelhart [4] empirically determined that the information capacity of a single product unit (as measured by its capacity for learning random boolean patterns) is approximately $3N$, compared to $2N$ for a single summation unit [2], where N is the number of inputs to the units. So, for a given problem, PUNNs usually require relatively smaller scale of networks than traditional feed-forward neural networks and even some feed-forward networks with other higher-order terms (cf. [12]). In [19], it has been demonstrated that only one product unit in the hidden layer is sufficient to solve the difficult symmetry problem and parity problem.

Fig. 1 shows the PUNN structure considered in this paper. It is a three-layer network with product units only in the hidden layer and only one output node.

Let the numbers of input and hidden nodes be N and $M+1$ (including a bias unit with fixed output -1), respectively. Write $\mathbf{w}_m = (w_{m,1}, \dots, w_{m,N})^{\mathrm{T}} \in \mathbb{R}^N$ $(1 \leqslant m \leqslant M)$ as the input-to-hidden weight vector corresponding to the mth hidden node. Similarly write $\mathbf{w}_{M+1} = (w_{M+1,0}, w_{M+1,1}, \dots, w_{M+1,M})^{\mathrm{T}} \in \mathbb{R}^{M+1}$ as the hidden-to-output weight vector, where $w_{M+1,0}$ is the threshold. For simplicity, all the weight vectors are incorporated into a total weight vector $\mathbf{W} = ((\mathbf{w}_1)^{\mathrm{T}}, \dots, (\mathbf{w}_{M+1})^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{MN+M+1}$. Let a vector $\mathbf{x} = (x_1, \dots, x_N)^{\mathrm{T}}$ with nonzero components be an input to the network shown in Fig. 1, with $x_1 = -1$ corresponding to a bias unit, the complex output of the network is

$$y = g\left(\left(\sum_{m=1}^{M} w_{M+1,m} h\left(\prod_{n=1}^{N} x_n^{w_{m,n}}\right)\right) - w_{M+1,0}\right). \tag{1}$$

Here, g is an activation function which squashes the output of the summation unit. All the outputs of the product units are squashed by an activation function h. A typical choice is $h(t) = t$. So we have

$$y = g\left(\left(\sum_{m=1}^{M} w_{M+1,m} \prod_{n=1}^{N} x_n^{w_{m,n}}\right) - w_{M+1,0}\right). \tag{2}$$

We stress that as in e.g. [10,19], we have required that $x_n \neq 0$ for $1 \leqslant n \leqslant N$. The complex output of the mth $(1 \leqslant m \leqslant M)$ hidden node is

$$\prod_{n=1}^{N} x_n^{w_{m,n}} = \mathrm{e}^{\rho_m}(\cos(\pi\theta_m) + \mathrm{i}\,\sin(\pi\theta_m)),$$

where

$$\rho_m = \sum_{n=1}^{N} w_{m,n} \ln|x_n|, \quad \theta_m = \sum_{n=1}^{N} w_{m,n}\sigma_n,$$
$$\sigma_n = \begin{cases} 0 & \text{if } x_n > 0; \\ 1 & \text{if } x_n < 0. \end{cases} \tag{3}$$

Durbin and Rumelhart [10] have discovered in their experiments that apart for the added complexity of working in the complex domain (i.e. the doubling of equations and weight variables), no substantial improvements in the results were gained. So the imaginary part of the output is omitted and the actual output for the mth hidden unit becomes $\mathrm{e}^{\rho_m}\cos(\pi\theta_m)$. But we mention a warning given by [18] that omitting the imaginary part of the output might cause trouble when this strategy was extended to real-valued (rather than Boolean) inputs.

For a given input $\mathbf{x} \in \{\mathbb{R}\backslash\{0\}\}^N$, define a multivariate function $f : \mathbb{R}^N \times \{\mathbb{R}\backslash\{0\}\}^N \to \mathbb{R}$ by

$$f(\mathbf{w}_M, \mathbf{x}) = \mathrm{e}^{\rho_m}\cos(\pi\theta_m). \tag{4}$$

It is obvious that $f \in C^\infty$. Next, denote by $\mathbf{F}$ the output vector of the hidden layer with respect to a given input $\mathbf{x}$, then we have

$$\mathbf{F} = (-1, f(\mathbf{w}_1, \mathbf{x}), \dots, f(\mathbf{w}_M, \mathbf{x}))^{\mathrm{T}}. \tag{5}$$

Consequently, the *actual output* of the PUNN for a given input $\mathbf{x}$ with nonzero components is the real part of the complex output as follows:

$$\begin{aligned} y &= g(\mathbf{w}_{M+1} \cdot \mathbf{F}) \\ &= g\left(\left(\sum_{m=1}^{M} w_{M+1,m} f(\mathbf{w}_M, \mathbf{x})\right) - w_{M+1,0}\right). \end{aligned} \tag{6}$$

3. BP algorithm for PUNN training

Let the network be supplied with a given set of training examples $\{\mathbf{x}^j, O^j\}_{j=1}^J \subset \{\mathbb{R}\backslash\{0\}\}^N \times \mathbb{R}$. For each $\mathbf{x}^j$, once

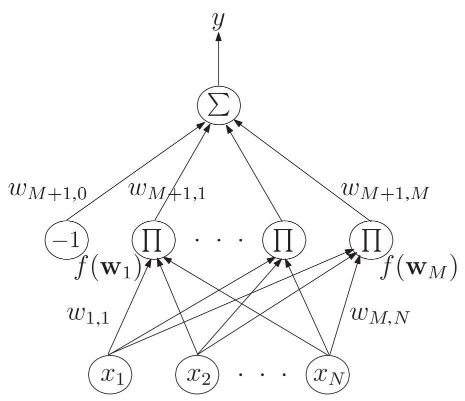


Fig. 1. Structure of PUNN.

being input into the above PUNN, the output of the PUNN above is

$$y^j = g(\mathbf{w}_{M+1}\cdot\mathbf{F}^j) = g\left(\left(\sum_{m=1}^{M} w_{M+1,m} f^j(\mathbf{w}_m)\right) - w_{M+1,0}\right), \tag{7}$$

where

$$f^j(\mathbf{w}) = f(\mathbf{w},\mathbf{x}^j), \quad \mathbf{F}^j = (-1, f^j(\mathbf{w}_1),\ldots,f^j(\mathbf{w}_M))^{\mathrm{T}}. \tag{8}$$

The square error function of PUNN trained by the BP algorithm can be represented as follows:

$$E(\mathbf{W}) = \frac{1}{2}\sum_{j=1}^{J}(y^j - O^j)^2 = \sum_{j=1}^{J} g_j(\mathbf{w}_{M+1}\cdot\mathbf{F}^j), \tag{9}$$

where

$$g_j(t) = \tfrac{1}{2}(g(t) - O^j)^2, \quad t\in\mathbb{R},\ 1\leqslant j\leqslant J. \tag{10}$$

So we have

$$\frac{\partial E}{\partial \mathbf{w}_{M+1}}(\mathbf{W}) = \sum_{j=1}^{J} g_j'(\mathbf{w}_{M+1}\cdot\mathbf{F}^j)\mathbf{F}^j \tag{11}$$

and

$$\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}) = \sum_{j=1}^{J} g_j'(\mathbf{w}_{M+1}\cdot\mathbf{F}^j)\frac{\partial}{\partial \mathbf{w}_m}(\mathbf{w}_{M+1}\cdot\mathbf{F}^j) = \sum_{j=1}^{J} g_j'(\mathbf{w}_{M+1}\cdot\mathbf{F}^j) w_{M+1,m}\frac{\partial f^j}{\partial \mathbf{w}}(\mathbf{w}_m), \tag{12}$$

where, for any $1\leqslant j\leqslant J$ and $1\leqslant m\leqslant M$,

$$\frac{\partial f^j}{\partial \mathbf{w}}(\mathbf{w}_m) = \left(\frac{\partial f^j}{\partial w_1}(\mathbf{w}_m),\ldots,\frac{\partial f^j}{\partial w_N}(\mathbf{w}_m)\right)^{\mathrm{T}}, \tag{13}$$

$$\frac{\partial f^j}{\partial w_n}(\mathbf{w}_m) = \mathrm{e}^{\rho_m^j}((\ln|x_n^j|)\cos(\pi\theta_m^j) - \sigma_n^j\pi\sin(\pi\theta_m^j)). \tag{14}$$

The purpose of the network training is to find $\mathbf{W}^*$ such that

$$E(\mathbf{W}^*) = \min E(\mathbf{W}).$$

Starting from an arbitrary initial value $\mathbf{W}^0$ and combining (11) and (12), we proceed to refine the weights by the training iteration as follows:

$$\mathbf{W}^{k+1} = \mathbf{W}^k + \Delta\mathbf{W}^k, \quad k = 0,1,2,\ldots, \tag{15}$$

where
$\Delta\mathbf{W}^k = ((\Delta\mathbf{w}_1^k)^{\mathrm{T}},\ldots,(\Delta\mathbf{w}_{M+1}^k)^{\mathrm{T}})^{\mathrm{T}} = -\eta(\partial E/\partial\mathbf{W})(\mathbf{W}^k)$, $\eta > 0$ is the learning rate, and

$$\Delta\mathbf{w}_{M+1}^k = -\eta\frac{\partial E}{\partial \mathbf{w}_{M+1}}(\mathbf{W}^k) = -\eta\sum_{j=1}^{J} g_j'(\mathbf{w}_{M+1}^k\cdot\mathbf{F}^j)\mathbf{F}^j, \tag{16}$$

$$\Delta\mathbf{w}_m^k = -\eta\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k) = -\eta\sum_{j=1}^{J} g_j'(\mathbf{w}_{M+1}^k\cdot\mathbf{F}^j) w_{M+1,m}^k\frac{\partial f^j}{\partial \mathbf{w}}(\mathbf{w}_m^k), \tag{17}$$

where $1\leqslant m\leqslant M$.

4. Convergence result

We need the following assumptions:

(A1) there exists a constant $C_1 > 0$ such that for all $t\in\mathbb{R}$

$$\max\{|g(t)|, |g'(t)|, |g''(t)|\}\leqslant C_1;$$

(A2) there exists a constant $C_2 > 0$ such that $\|\mathbf{w}_m^k\|\leqslant C_2$ for all $m = 1,\ldots,M+1$, $k = 0,1,2,\ldots$;
(A3) the learning rate η satisfies (A.18);
(A4) the set $\mathbf{D}_0 = \{\mathbf{W}|(\partial E/\partial\mathbf{W})(\mathbf{W}) = 0\}$ contains finite points.

In this paper, $\|\cdot\|$ denotes the Euclidean norm.

Theorem 1. *Let the error function* $E(\mathbf{W})$ *be defined in* (9), *and the sequence* $\{\mathbf{W}^k\}$ *be generated by* (15)–(17) *with* $\mathbf{W}^0$ *being an arbitrary initial guess. If Assumptions* (A1)–(A3) *are valid, then we have*

(i) $E(\mathbf{W}^{k+1})\leqslant E(\mathbf{W}^k)$, $k = 0,1,2,\ldots$;
(ii) $\lim_{k\to\infty}\|(\partial E/\partial\mathbf{W})(\mathbf{W}^k)\| = 0$.
Furthermore, if Assumption (A4) *also holds, there exists a point* $\mathbf{W}^*\in\mathbf{D}_0$ *such that*
(iii) $\lim_{k\to\infty}\mathbf{W}^k = \mathbf{W}^*$.

The statement (i) of Theorem 1 shows the monotonicity of error function $E(\mathbf{W})$ in the learning iteration process. The weak convergence of the weight sequence $\{\mathbf{W}^k\}$ is presented in (ii), i.e., starting from arbitrary $\mathbf{W}^0$ and generated in accordance with (15)–(17), the sequence $\{\mathbf{W}^k\}$ will satisfy $\lim_{k\to\infty}\|(\partial E/\partial\mathbf{W})(\mathbf{W}^k)\| = 0$. The conclusion (iii) points out that if the number of stationary points is finite, the sequence $\{\mathbf{W}^k\}$ will converge to a local minimum, which implies the feasibility of the combination of a global search algorithm and BP. If there are finite number of stationary points (i.e. local minima) in the search space, the neighborhood of the global minimum may be detected by the global search algorithm (e.g. RSA), and then the global minimum $\mathbf{W}^*$ will be located by BP.

5. Experimental results

This section illustrates the performance of BP for training PUNNs. The 5-Parity problem is used as the test problem. This problem can be solved theoretically by a single product unit (cf. [19]), so we can easily compare the theoretical solution with the numerical results.

The network has six inputs ($N = 6$ in Fig. 1) including a constant input of -1 (the bias unit), two hidden product units ($M = 2$) plus a bias unit, and one summation output unit. The learning rate is fixed to 0.05 for all the simulation.

First, we check the convergence for an arbitrary initial weight, i.e., starting from an arbitrary guess, the weight sequence generated by (15)–(17) satisfy $E(\mathbf{W}^{k+1}) \leqslant E(\mathbf{W}^k)$ and $\lim_{k\to\infty}\|(\partial E/\partial \mathbf{W})(\mathbf{W}^k)\| = 0$. A typical picture of such a learning iteration process is given in Fig. 2, which clearly supports the theoretical convergence result, namely, the weight sequence will always converge to a local minimum of the error function.

Next, we investigate the convergence near the global minimum of the error function, that is, the optimal weights of the network. As in [19], we can find explicitly the optimal weights of the network for the 5-Parity problem: $\mathbf{w}_1 = (1,1,1,1,-1)^{\mathrm{T}}$, $\mathbf{w}_2 = (0,0,0,0,0)^{\mathrm{T}}$ and $\mathbf{w}_3 = (0,5,0)^{\mathrm{T}}$. Then we randomly initialize the weights within a neighborhood, with 0.05 radius, of the optimal weights. Fig. 3 shows that, in spite of the oscillation in the early epochs, both the error curve and the derivative curve finally tend to zero.

The weight sequence will converge to a local minimum with the initial weight being an arbitrary guess. In particular, when the initial weight falls into a small neighborhood of a local (or global) minimum, the weight sequence will converge quickly to the local (or global) minimum. This supports our idea that, if a neighborhood of the local (or global) minimum is detected through some other approaches, the corresponding local (or global) minimum will be located efficiently by BP algorithm.

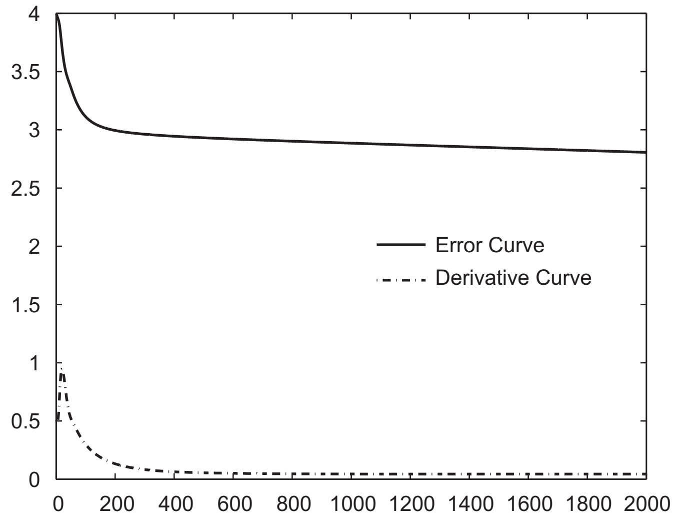


Fig. 2. Convergence behavior from an initial arbitrary initial weight.

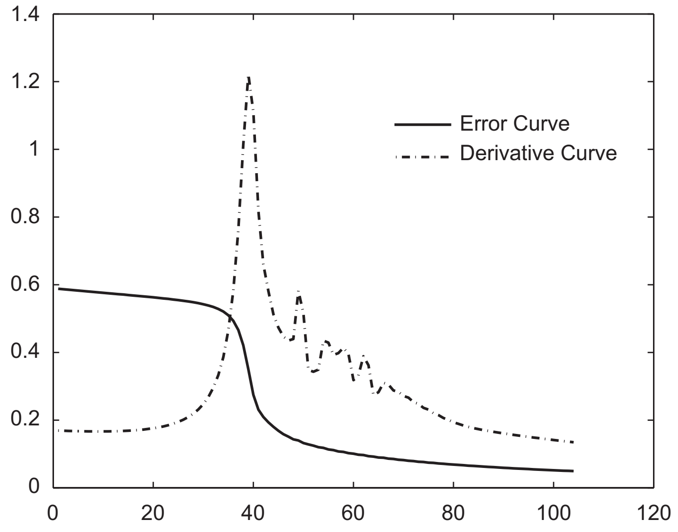


Fig. 3. Convergence behavior from an initial weight near the global minimum.

6. Conclusion

Product unit neural networks with exponential weights (PUNNs) were proposed in order to improve the efficiency of traditional feed-forward neural networks [4]. Unfortunately, the solution space for PUNNs can be extremely convoluted, with numerous local minima that trap the BP training. Hence, some global optimization algorithms such as Genetic Algorithm, etc. have been used to train PUNNs. But these global optimization algorithms usually converge very slowly. So it seems natural that BP algorithm, which is good at local optimization, has been used to combine with some global search algorithms, resulting in quite satisfactory training. So the convergence analysis of the BP algorithm for PUNN is still useful as a theoretical support of this training strategy.

In this paper, a convergence result of BP algorithm for training PUNNs is presented. The monotonicity of the error function in the training iteration process is also guaranteed. An up-bound of the learning rate η is provided to guarantee the monotonicity and the convergence. A numerical example is given to support the theoretical findings.

We mention that the convergence results in this paper can be easily extended to a more general case that the network has several outputs. Also, it seems promising to apply the idea of PUNN to some more complex and capable neural networks such as stochastic neural networks and neural networks with delays (cf. [13,20]).

Appendix A

In this appendix, we first present some lemmas and then prove Theorem 1.

The Euclidean norm of a vector $\mathbf{w} = (w_1, \ldots, w_N)^{\mathrm{T}} \in \mathbb{R}^N$ is defined by $\|\mathbf{w}\| = (\sum_{n=1}^{N} w_n^2)^{1/2}$, and the corresponding induced norm of a matrix $\mathbf{H} \in \mathbb{R}^{N\times N}$ by $\|\mathbf{H}\| = \sup_{\mathbf{w}\neq 0}(\|\mathbf{H}\mathbf{w}\|/\|\mathbf{w}\|)$. The following properties of the norms are well-known: $\|\mathbf{H}\mathbf{w}\| \leqslant \|\mathbf{H}\|\,\|\mathbf{w}\|$, $|\mathbf{w}^{\mathrm{T}}\mathbf{H}\mathbf{w}| \leqslant \|\mathbf{w}\|\,\|\mathbf{H}\|\,\|\mathbf{w}\|$.

Lemma 2. *Suppose that* $E:\mathbb{R}^{MN+M+1}\to\mathbb{R}$ *is continuous and differentiable on a compact set* $\mathbf{D}\subset\mathbb{R}^{MN+M+1}$, *and that* $\mathbf{D}_0=\{\mathbf{W}\in\mathbf{D}|(\partial E/\partial\mathbf{W})(\mathbf{W})=0\}$ *has only finite number of points. If a sequence* $\{\mathbf{W}^k\}_{k=1}^{\infty}\subset\mathbf{D}$ *satisfies*

$$\lim_{k\to\infty}\|\mathbf{W}^{k+1}-\mathbf{W}^k\|=0,\quad \lim_{k\to\infty}\left\|\frac{\partial E}{\partial\mathbf{W}}(\mathbf{W}^k)\right\|=0,$$

then there exists a point $\mathbf{W}^*\in\mathbf{D}_0$ *such that*

$$\lim_{k\to\infty}\mathbf{W}^k=\mathbf{W}^*.$$

Proof. This result is basically the same as Theorem 14.1.5 in [24] (cf. [23]), and the detailed proof is thus omitted. □

For any $1\leqslant j\leqslant J$, $1\leqslant m\leqslant M$ and $k=0,1,2,\ldots,$ write (cf. (8))

$$f_m^{k,j}=f^j(\mathbf{w}_m^k),\quad \mathbf{F}^{k,j}=(-1,f_1^{k,j},\ldots,f_M^{k,j})^{\mathrm{T}},$$
$$\boldsymbol{\psi}^{k,j}=\mathbf{F}^{k+1,j}-\mathbf{F}^{k,j},\quad \phi^{k,j}=\mathbf{w}_{M+1}^k\cdot\mathbf{F}^{k,j}. \tag{A.1}$$

Denote the Hessian matrix of $f^j(\mathbf{w})$ (cf. (8)) by

$$\mathbf{H}^j(\mathbf{w})=\begin{pmatrix}\dfrac{\partial^2 f^j}{\partial w_1\,\partial w_1}(\mathbf{w}) & \cdots & \dfrac{\partial^2 f^j}{\partial w_1\,\partial w_N}(\mathbf{w})\\ \vdots & \ddots & \vdots\\ \dfrac{\partial^2 f^j}{\partial w_N\,\partial w_1}(\mathbf{w}) & \cdots & \dfrac{\partial^2 f^j}{\partial w_N\,\partial w_N}(\mathbf{w})\end{pmatrix}. \tag{A.2}$$

$\mathbf{H}^j(\mathbf{w})$ is symmetric, since $f^j(\mathbf{w})\in C^\infty$. Specifically, for any $p,q\in\{1,\ldots,N\}$,

$$\begin{aligned}&\frac{\partial^2 f^j}{\partial w_p\,\partial w_q}(\mathbf{w})\\ &\quad= e^{\sum_{n=1}^N w_n\ln|x_n^j|}\Bigg\{[(\ln|x_p^j|)(\ln|x_q^j|)-\sigma_p^j\sigma_q^j\pi^2]\cos\left(\pi\sum_{n=1}^N w_n\sigma_n^j\right)\\ &\qquad-[\pi\sigma_q^j(\ln|x_p^j|)+\pi\sigma_p^j(\ln|x_q^j|)]\sin\left(\pi\sum_{n=1}^N w_n\sigma_n^j\right)\Bigg\},\end{aligned} \tag{A.3}$$

where σ_n^j is defined similarly as in (3).

Lemma 3. *Suppose Assumptions* (A1)–(A2) *hold, then for any* $1\leqslant j\leqslant J$ *and* $k=0,1,2,\ldots,$ *we have*

$$\|O^j\|\leqslant C_0,\quad \|\mathbf{F}^{k,j}\|\leqslant C_0, \tag{A.4}$$

$$\left\|\frac{\partial f^j}{\partial\mathbf{w}}(\mathbf{w})\right\|\leqslant C_3,\quad \|\mathbf{H}^j(\mathbf{w})\|\leqslant C_3,\quad \forall\mathbf{w}\in\{\mathbf{w}\in\mathbb{R}^N\,|\,\|\mathbf{w}\|\leqslant C_2\}, \tag{A.5}$$

$$|g_j'(t)|\leqslant C_4,\quad |g_j''(t)|\leqslant C_4,\quad t\in\mathbb{R}, \tag{A.6}$$

$$\|\boldsymbol{\psi}^{k,j}\|^2\leqslant C_5\sum_{m=1}^M\|\Delta\mathbf{w}_m^k\|^2, \tag{A.7}$$

$$\sum_{j=1}^J g_j'(\phi^{k,j})(\Delta\mathbf{w}_{M+1}^k\cdot\mathbf{F}^{k,j})=-\eta\left\|\frac{\partial E}{\partial\mathbf{w}_{M+1}}(\mathbf{W}^k)\right\|^2, \tag{A.8}$$

$$\sum_{j=1}^J g_j'(\phi^{k,j})(\mathbf{w}_{M+1}^k\cdot\boldsymbol{\psi}^{k,j})\leqslant-\eta\sum_{m=1}^M\left\|\frac{\partial E}{\partial\mathbf{w}_m}(\mathbf{W}^k)\right\|^2+C_6\eta^2\sum_{m=1}^M\left\|\frac{\partial E}{\partial\mathbf{w}_m}(\mathbf{W}^k)\right\|^2, \tag{A.9}$$

$$\sum_{j=1}^J g_j'(\phi^{k,j})\Delta\mathbf{w}_{M+1}^k\cdot\boldsymbol{\psi}^{k,j}\leqslant C_7\eta^2\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial\mathbf{w}_m}(\mathbf{W}^k)\right\|^2, \tag{A.10}$$

$$\frac12\sum_{j=1}^J g_j''(t^{k,j})(\phi^{k+1,j}-\phi^{k,j})^2\leqslant C_8\eta^2\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial\mathbf{w}_m}(\mathbf{W}^k)\right\|^2, \tag{A.11}$$

$$\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial\mathbf{w}_m}(\mathbf{W}^k)\right\|^2=\left\|\frac{\partial E}{\partial\mathbf{W}}(\mathbf{W}^k)\right\|^2, \tag{A.12}$$

where C_i $(i=0,3,\ldots,8)$ *are constants independent of* k *and* j, *and each* $t^{k,j}\in\mathbb{R}$ *lies on the segment between* $\phi^{k,j}$ *and* $\phi^{k+1,j}$.

Proof. If the set of samples is fixed and Assumption (A2) is valid, according to (8), (13)–(14) and (A.1)–(A.3), the validations of (A.4) and (A.5) can be easily got.

By (10), for any $1\leqslant j\leqslant J$ and $t\in\mathbb{R}$,

$$g_j'(t)=g'(t)(g(t)-O^j),$$
$$g_j''(t)=g''(t)(g(t)-O^j)+(g'(t))^2.$$

Then, (A.6) follows directly from Assumption (A1) and $C_4=C_1(C_1+C_0)+(C_1)^2$.

It follows from (A.1), (A.5), the Cauchy–Schwartz Inequality and the Mean-Value Theorem for multivariate functions [1] that for any $1\leqslant j\leqslant J$ and $k=0,1,2,\ldots,$

$$\begin{aligned}\|\boldsymbol{\psi}^{k,j}\|^2&=\|\mathbf{F}^{k+1,j}-\mathbf{F}^{k,j}\|^2\\ &=\left\|\begin{pmatrix}0\\ f^j(\mathbf{w}_1^{k+1})-f^j(\mathbf{w}_1^k)\\ \vdots\\ f^j(\mathbf{w}_M^{k+1})-f^j(\mathbf{w}_M^k)\end{pmatrix}\right\|^2\\ &=\left\|\begin{pmatrix}0\\ \dfrac{\partial f^j}{\partial\mathbf{w}}(\mathbf{s}_1^{k,j})\cdot\Delta\mathbf{w}_1^k\\ \vdots\\ \dfrac{\partial f^j}{\partial\mathbf{w}}(\mathbf{s}_M^{k,j})\cdot\Delta\mathbf{w}_M^k\end{pmatrix}\right\|^2\\ &=\sum_{m=1}^M\left(\left|\frac{\partial f^j}{\partial\mathbf{w}}(\mathbf{s}_m^{k,j})\cdot\Delta\mathbf{w}_m^k\right|\right)^2\leqslant C_5\sum_{m=1}^M\|\Delta\mathbf{w}_m^k\|^2,\end{aligned}$$

where $C_5=(C_3)^2$ and $\mathbf{s}_m^{k,j}$ is an intermediate point on the line segment between the two points $\mathbf{w}_m^{k+1}$ and $\mathbf{w}_m^k$.

According to (11) and (16), for any $k = 0, 1, 2, \ldots$, we can get

$$\begin{aligned}\sum_{j=1}^{J} g_j'(\phi^{k,j})(\Delta\mathbf{w}_{M+1}^k \cdot \mathbf{F}^{k,j}) &= \sum_{j=1}^{J} g_j'(\phi^{k,j})(\mathbf{F}^{k,j})^{\mathrm{T}}\Delta\mathbf{w}_{M+1}^k \\ &= \frac{\partial E}{\partial \mathbf{w}_{M+1}}(\mathbf{W}^k)\cdot\Delta\mathbf{w}_{M+1}^k \\ &= -\eta\left\|\frac{\partial E}{\partial \mathbf{w}_{M+1}}(\mathbf{W}^k)\right\|^2.\end{aligned}$$

Next, we prove (A.9). Using (13), (A.1) and the Taylor expansion for multivariate functions [1], for any $1 \leqslant j \leqslant J$, $1 \leqslant m \leqslant M$ and $k = 0, 1, 2, \ldots$, we have that

$$\begin{aligned}f_m^{k+1,j} - f_m^{k,j} = {}& \left(\frac{\partial f^j}{\partial \mathbf{w}}(\mathbf{w}_m^k)\right)^{\mathrm{T}}\Delta\mathbf{w}_m^k \\ &+ \frac{1}{2}(\Delta\mathbf{w}_m^k)^{\mathrm{T}}\mathbf{H}^j(\mathbf{s}_m^{k,j})(\Delta\mathbf{w}_m^k),\end{aligned} \tag{A.13}$$

where $\mathbf{s}_m^{k,j}$ is an intermediate point on the line segment between the two points $\mathbf{w}_m^{k+1}$ and $\mathbf{w}_m^k$. Then, in terms of (12), (17), (A.1) and (A.13), for any $k = 0, 1, 2, \ldots$, we get that

$$\begin{aligned}&\sum_{j=1}^{J} g_j'(\phi^{k,j})(\mathbf{w}_{M+1}^k \cdot \boldsymbol{\psi}^{k,j}) \\ &= \sum_{j=1}^{J}\sum_{m=1}^{M} g_j'(\phi^{k,j}) w_{M+1,m}^k (f_m^{k+1,j} - f_m^{k,j}) \\ &= \sum_{j=1}^{J}\sum_{m=1}^{M} g_j'(\mathbf{w}_{M+1}^k \cdot \boldsymbol{\psi}^{k,j}) w_{M+1,m}^k \\ &\quad \times \left(\frac{\partial f^j}{\partial \mathbf{w}}(\mathbf{w}_m^k)\right)^{\mathrm{T}}\Delta\mathbf{w}_m^k + \delta_1 \\ &= \sum_{m=1}^{M}\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\cdot\Delta\mathbf{w}_m^k + \delta_1 \\ &= -\eta\sum_{m=1}^{M}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2 + \delta_1,\end{aligned} \tag{A.14}$$

where

$$\begin{aligned}\delta_1 = {}& \frac{1}{2}\sum_{j=1}^{J}\sum_{m=1}^{M} g_j'(\mathbf{w}_{M+1}^k \cdot \boldsymbol{\psi}^{k,j}) \\ &\times w_{M+1,m}^k(\Delta\mathbf{w}_m^k)^{\mathrm{T}}\mathbf{H}^j(\mathbf{s}_m^{k,j})\Delta\mathbf{w}_m^k.\end{aligned}$$

A combination of Assumptions (A2), (A.5) and (A.6) leads to

$$\delta_1 \leqslant C_6\sum_{m=1}^{M}\|\Delta\mathbf{w}_m^k\|^2 = C_6\eta^2\sum_{m=1}^{M}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2, \tag{A.15}$$

where $C_6 = (J/2)C_2C_3C_4$. Finally, (A.9) results from (A.14) and (A.15).

According to (A.1), (A.6), (A.7) and the Cauchy–Schwartz Inequality, we have that

$$\begin{aligned}&\sum_{j=1}^{J} g_j'(\phi^{k,j})\Delta\mathbf{w}_{M+1}^k \cdot \boldsymbol{\psi}^{k,j} \\ &\leqslant C_4\sum_{j=1}^{J}\|\Delta\mathbf{w}_{M+1}^k\|\|\boldsymbol{\psi}^{k,j}\| \\ &\leqslant \frac{1}{2}C_4\sum_{j=1}^{J}(\|\Delta\mathbf{w}_{M+1}^k\|^2 + \|\boldsymbol{\psi}^{k,j}\|^2) \\ &\leqslant \frac{J}{2}C_4\left(\|\Delta\mathbf{w}_{M+1}^k\|^2 + C_5\sum_{m=1}^{M}\|\Delta\mathbf{w}_m^k\|^2\right) \\ &\leqslant C_7\eta^2\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2\end{aligned}$$

and

$$\begin{aligned}&\frac{1}{2}\sum_{j=1}^{J} g_j''(t^{k,j})(\phi^{k+1,j} - \phi^{k,j})^2 \\ &\leqslant \frac{C_4}{2}\sum_{j=1}^{J}(\phi^{k+1,j} - \phi^{k,j})^2 \\ &= \frac{C_4}{2}\sum_{j=1}^{J}((\mathbf{w}_{M+1}^{k+1} - \mathbf{w}_{M+1}^k)\cdot\mathbf{F}^{k+1,j} \\ &\quad + \mathbf{w}_{M+1}^k\cdot(\mathbf{F}^{k+1,j} - \mathbf{F}^{k,j}))^2 \\ &\leqslant \frac{C_4}{2}\max\{(C_0)^2, (C_2)^2\} \\ &\quad \times \sum_{j=1}^{J}\left(\|\Delta\mathbf{w}_{M+1}^k\| + \|\boldsymbol{\psi}^{k,j}\|\right)^2 \\ &\leqslant C_8\left(\|\Delta\mathbf{w}_{M+1}^k\|^2 + \sum_{m=1}^{M}\|\Delta\mathbf{w}_m^k\|^2\right) \\ &\leqslant C_8\eta^2\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2,\end{aligned}$$

where $C_7 = (C_4J/2)\max\{1, C_5\}$ and $C_8 = (C_4J/2)\max\{(C_0)^2, (C_2)^2\}\max\{1, C_5\}$. So we obtain (A.10) and (A.11).

It follows from the definition of Euclidean norm that

$$\begin{aligned}&\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2 \\ &= \sum_{m=1}^{M+1}\left(\left(\frac{\partial E(\mathbf{W}^k)}{\partial w_{m,1}}\right)^2 + \cdots + \left(\frac{\partial E(\mathbf{W}^k)}{\partial w_{m,N}}\right)^2\right) \\ &= \sum_{m=1}^{M+1}\sum_{n=1}^{N}\left(\frac{\partial E(\mathbf{W}^k)}{\partial w_{m,n}}\right)^2 = \left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^k)\right\|^2.\end{aligned}$$

This confirms (A.12) and completes the proof. □

Now, we use the above two lemmas to prove Theorem 1.

Proof of Theorem 1. (i) By (A.8)–(A.12) and the Taylor Expansion we have

$$
\begin{aligned}
&E(\mathbf{W}^{k+1}) - E(\mathbf{W}^k)\\
&\quad = \sum_{j=1}^{J}(g_j(\phi^{k+1,j}) - g_j(\phi^{k,j}))\\
&\quad = \sum_{j=1}^{J}\left(g_j'(\phi^{k,j})(\phi^{k+1,j} - \phi^{k,j}) + \frac{1}{2}g_j''(t^{k,j})(\phi^{k+1,j} - \phi^{k,j})^2\right)\\
&\quad = \sum_{j=1}^{J} g_j'(\phi^{k,j})(\Delta\mathbf{w}_{M+1}^k \cdot \mathbf{F}^{k,j} + \mathbf{w}_{M+1}^k \cdot \boldsymbol{\psi}^{k,j} + \Delta\mathbf{w}_{M+1}^k \cdot \boldsymbol{\psi}^{k,j})\\
&\qquad + \frac{1}{2}\sum_{j=1}^{J} g_j''(t^{k,j})(\phi^{k+1,j} - \phi^{k,j})^2 \leqslant -\eta\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2\\
&\qquad + C_6\eta^2\sum_{m=1}^{M}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2 + (C_7 + C_8)\eta^2\sum_{m=1}^{M+1}\left\|\frac{\partial E}{\partial \mathbf{w}_m}(\mathbf{W}^k)\right\|^2\\
&\qquad \leqslant -(\eta - C_9\eta^2)\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^k)\right\|^2,
\end{aligned}
\tag{A.16}
$$

where $C_9 = C_6 + C_7 + C_8$ and $t^{k,j} \in \mathbb{R}$ is on the segment between $\phi^{k,j}$ and $\phi^{k+1,j}$. Let $\beta = \eta - C_9\eta^2$, then we have

$$
E(\mathbf{W}^{k+1}) \leqslant E(\mathbf{W}^k) - \beta\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^k)\right\|^2. \tag{A.17}
$$

Obviously, if the learning rate η is chosen to satisfy that

$$
0 < \eta < \frac{1}{C_9}, \tag{A.18}
$$

then there will hold that

$$
E(\mathbf{W}^{k+1}) \leqslant E(\mathbf{W}^k), \quad k = 0, 1, 2, \ldots .
$$

(ii) Using (A.17), we have that

$$
\begin{aligned}
E(\mathbf{W}^{k+1}) &\leqslant E(\mathbf{W}^k) - \beta\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^k)\right\|^2\\
&\leqslant \cdots \leqslant E(\mathbf{W}^0) - \beta\sum_{t=0}^{k}\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^t)\right\|^2.
\end{aligned}
$$

Since $E(\mathbf{W}^{k+1}) \geqslant 0$, there holds that

$$
\beta\sum_{t=0}^{k}\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^t)\right\|^2 \leqslant E(\mathbf{W}^0).
$$

Let $k \to \infty$, then

$$
\sum_{t=0}^{\infty}\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^t)\right\|^2 \leqslant E(\mathbf{W}^0) < \infty.
$$

So there holds that

$$
\lim_{k\to\infty}\left\|\frac{\partial E}{\partial \mathbf{W}}(\mathbf{W}^k)\right\| = 0.
$$

(iii) It follows from (15)–(17) and (ii) in Theorem 1 that

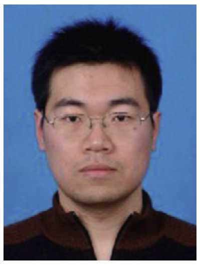
$$
\lim_{k\to\infty}\|\Delta\mathbf{W}^k\| = 0. \tag{A.19}
$$

Note that the error function $E(\mathbf{W})$ defined in (9) is continuously differentiable. Using (A.19), Assumption (A4) and Lemma 2, we immediately get the desired result. This completes the proof to Theorem 1. □

References

[1] R. Courant, F. John, Introduction to Calculus and Analysis, Wiley, New York, 1974.
[2] T. Cover, Geometrical and statistical properties of systems of linear inequalities with application in pattern recognition, IEEE Trans. Electron. Comput. 14 (1965) 326–334.
[3] G. Cybenko, Continuous-valued neural networks with two hidden layers are sufficient, Technical Report, Department of Computer Science, Tufts University, Medford, MA, 1989.
[4] R. Durbin, D. Rumelhart, Product units: a computationally powerful and biologically plausible extension to backpropagation networks, Neural Comput. 1 (1989) 133–142.
[5] A.P. Engelbrecht, Computational Intelligence: An Introduction, Wiley, New York, 2003.
[6] C.L. Giles, Learning, invariance, and generalization in higher-order neural networks, Appl. Opt. 26 (1987) 4972–4978.
[7] K.N. Gurney, Training nets of hardware realizable Sigma-Pi units, Neural Networks 5 (1992) 289–303.
[8] C. Hervš, F.J. Martänez-Estudillo, P.A. Gutiřrez, Classification by means evolutionary product-unit neural networks, in: International Joint Conference on Neural Networks (IJCNN 2006), 2006, pp. 1525–1532.
[9] K. Hornik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[10] A. Ismail, A.P. Engelbrecht, Global optimization algorithms for training product unit neural networks, in: The Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, vol. 1, 2000, pp. 132–137.
[11] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert Mag. 8 (1993) 26–33.
[12] L.R. Leerink, C.L. Giles, B.G. Horne, M.A. Jabri, Learning with product units, Adv. Neural Inform. Process. Syst. 7 (1995) 537.
[13] Y.R. Liu, Z.D. Wang, X.H. Liu, On global exponential stability of generalized stochastic neural networks with mixed time delays, Neurocomputing 70 (2006) 314–326.
[14] A.C. Martänez-Estudillo, C. Hervš-Martänez, F.J. Martänez-Estudillo, N. Garcäa-Pedrajas, Hybridation of evolutionary algorithms and local search by means of a clustering method, IEEE Trans. Syst. Man Cybern. Part. B Cybern. 36 (2006) 534–546.
[15] A.C. Martänez-Estudillo, F.J. Martänez-Estudillo, C. Hervš-Martänez, N. Garcäa-Pedrajas, Evolutionary product unit based neural networks for regression, Neural Networks 19 (2006) 477–486.
[16] K. Saito, R. Nakano, Law discovery using neural networks, in: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI97), 1997, pp. 1078–1083.
[17] K. Saito, R. Nakano, Extracting regression rules from neural networks, Neural Networks 15 (2002) 1279–1288.
[18] M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, Neural Comput. 14 (2001) 241–301.
[19] J.H. Wang, Y.W. Yu, J.H. Tsai, On the internal representations of product units, Neural Process. Lett. 12 (2000) 247–254.
[20] Z.D. Wang, Y.R. Liu, X.H. Liu, On global asymptotic stability of neural networks with discrete and distributed delays, Phys. Lett. A 345 (2005) 299–308.
[21] W. Wu, G. Feng, X. Li, Training multilayer perceptrons via minimization of sum of ridge functions, Adv. Comput. Math. 17 (2002) 331–347.
[22] W. Wu, G. Feng, Z. Li, Y. Xu, Convergence of an online gradient method for BP neural networks, IEEE Trans. Neural Networks 16 (2005) 533–540.

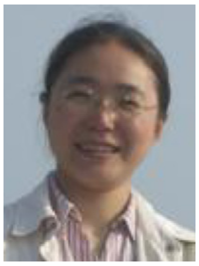
[23] W. Wu, H.M. Shao, D. Qu, Strong convergence for gradient methods for BP networks training, in: Proceedings of the International Conference on Neural Networks and Brains, 2005, pp. 332–334.
[24] Y. Yuan, W. Sun, Optimization Theory and Methods, Science Press, Beijing, 2001.



Chao Zhang was born in Dalian, PR China. He received B.S. in applied mathematics in 2004 and received M.S. degrees in computational mathematics in 2005, respectively, both from Dalian University of Technology. Now he is pursuing Ph.D. in computational mathematics in Dalian University of Technology. His research interests lie in the areas of neural networks and unconstrained optimization.

Wei Wu received the M.S. degree from Jilin University, Changchun, China, in 1981 and the D.Phil. degree from Oxford University, Oxford, UK, in 1987. He is now a professor of the Applied Mathematics Department of Dalian University of Technology, Dalian, China. He serves as associate editors for the Journal of Information & Computational Science, Numerical Mathematics—A Journal of Chinese Universities, and Journal of Mathematical Research and Exposition. His research interests include numerical analysis and neural network computation.

Xianhua Chen, borned in Heilongjiang province in 1984, has received B.S. degree in applied mathematics from Shandong Normal University in 2006. Now he is pursuing M.S. degree in computational mathematics in Dalian University of Technology. His current interest is focused on artificial neural networks and finite elements.



Yan Xiong received M.S. degree in Applied Mathematics with an emphasis on genetic algorithm from Northeastern University in 2003. Then she received Ph.D. in computational mathematics in Dalian University of Technology in 2007. Her research interests include the convergence analysis and structural optimization of neural networks, especially in higher order neural networks.