Neural Process Lett (2007) 26:177–189
DOI 10.1007/s11063-007-9050-0

Convergence Analysis of Batch Gradient Algorithm for Three Classes of Sigma-Pi Neural Networks

Chao Zhang · Wei Wu · Yan Xiong

Published online: 5 September 2007

Abstract Sigma-Pi (Σ-Π) neural networks (SPNNs) are known to provide more powerful mapping capability than traditional feed-forward neural networks. A unified convergence analysis for the batch gradient algorithm for SPNN learning is presented, covering three classes of SPNNs: Σ-Π-Σ, Σ-Σ-Π and Σ-Π-Σ-Π. The monotonicity of the error function in the iteration is also guaranteed.

Keywords Convergence · Sigma-Pi-Sigma neural networks · Sigma-Sigma-Pi neural networks · Sigma-Pi-Sigma-Pi neural networks · Batch gradient algorithm · Monotonicity

Mathematics Subject Classification (2000) 92B20 · 68Q32 · 74P05

Abbreviation:
SPNN Sigma-Pi neural network

1 Introduction

SPNNs may be configured into feed-forward neural networks that consist of Sigma-Pi (Σ-Π) units (cf. [1]). These networks are known to provide inherently more powerful mapping capability than traditional feed-forward networks with multiple layers of summation nodes in all the non-input layers [2,3]. The gradient algorithm is possibly the most popular optimization algorithm to train feed-forward neural networks [4,5]. Its convergence has been studied in e.g. [6–9] for traditional feed-forward neural networks. In this paper, we prove the convergence for the gradient learning methods for Sigma-Pi-Sigma neural networks. The proof is presented in a unified manner such that it also applies to other two classes of SPNNs, namely,

C. Zhang · W. Wu (✉) · Y. Xiong
Department of Applied Mathematics, Dalian University of Technology, Dalian 116024, P. R. China
e-mail: wuweiw@dlut.edu.cn

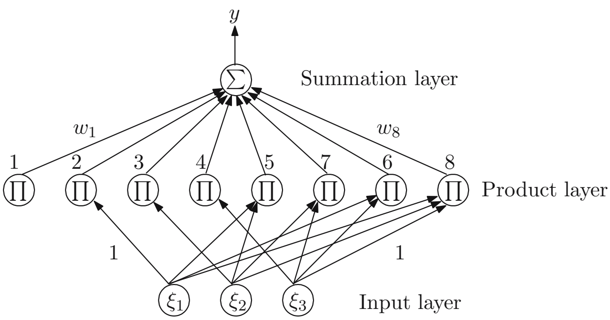

Fig. 1 A fully connected Sigma-Pi unit

Sigma-Sigma-Pi and Sigma-Pi-Sigma-Pi neural networks. It even applies to Sigma-Sigma neural networks, that is the ordinary feed-forward neural networks with a hidden layer.

The organization of the rest of this paper is as follows. Section 2 introduces the Sigma-Pi units, discusses the equivalence of the three classes of Sigma-Pi neural networks, and describes the working and learning procedures of Σ-Π-Σ neural networks. The main convergence results are presented in Sect. 3. Section 4 is an appendix, in which details of the proofs are provided.

2 Sigma-Pi Neural Networks

2.1 Sigma-Pi Units

A Sigma-Pi unit consists of an output layer with only one summation node, an input layer and a hidden layer of product nodes. The function of the product layer is to implement a polynomial expansion for the components of the input vector $\xi = (\xi_1, \xi_2, \dots, \xi_N)^T$. To do this, each product node is connected with certain nodes (say $\{1, 2\}$, $\{1, 3\}$, or $\{1, 2, 4\}$) of the input layer and corresponds to a particular monomial (say, correspondingly, $\xi_1\xi_2$, $\xi_1\xi_3$ or $\xi_1\xi_2\xi_4$). The N input nodes and the product nodes can be *fully connected* as shown in Fig. 1 with $N = 3$, with the number of the product nodes being $C_N^0 + C_N^1 + C_N^2 + \cdots + C_N^N = 2^N$ and the number of the weights between the input and product layers being $c_N = C_N^1 * 1 + C_N^2 * 2 + \cdots + C_N^N * N$. The N input nodes and the product nodes are *sparsely connected* if the number of the product nodes is less than 2^N and/or the number of the weights between the input and product layers is less that c_N. These monomials, i.e. the outputs of the product nodes, are used to form a weighted linear combination such as $w_1\xi_1\xi_2 + w_2\xi_1\xi_3 + w_3\xi_1\xi_2\xi_4 + \cdots$, by the operation of the summation layer.

Definition 1 Denote by N_P and N_I the numbers of nodes in the product and the input layers, respectively. Define Λ_i $(1 \le i \le N_P)$ as the set of the indexes of all the input nodes connected with the i-th product node, and V_j $(1 \le j \le N_I)$ the set of the indexes of all the product nodes connected with the j-th input node.

For example, in Fig. 1, the 1st product node, corresponding to the bias w_1, does not connect with any input node, so $\Lambda_1 = \varnothing$. And we have $\Lambda_3 = \{2\}$, $\Lambda_6 = \{2, 3\}$, $\Lambda_8 = \{1, 2, 3\}$, $V_1 = \{2, 5, 7, 8\}$, etc. We also note that $\Lambda_i \subseteq \{1, 2, \dots, N_I\}$ and $V_j \subseteq \{1, 2, \dots, N_P\}$. Different definitions of $\{\Lambda_i\}$ and $\{V_j\}$ result in different structures of a Sigma-Pi unit. For an arbitrary set A, let $\varphi(A)$ be the number of the elements in A. Then, we have

$$\sum_{i=1}^{N_P} \varphi(\Lambda_i) = \sum_{j=1}^{N_I} \varphi(V_j), \tag{1}$$

which will be used later in our proof.

We mention that arbitrary Boolean functions can be realized by a single fully connected Sigma-Pi unit, showing the great inherent power of Sigma-Pi units [2].

2.2 Equivalence of Sigma-Pi Neural Networks

Sigma-Pi unit can be used as a building block to construct many kinds of SPNNs: Σ-Π, Σ-Π-Σ, Σ-Σ-Π and Σ-Π-Σ-Π (cf. [1], [10], [11] and [12], respectively.), etc., where Σ and Π stand for a summation layer and a product layer, respectively. The Sigma-Pi unit shown in Fig. 1 is actually a special Σ-Π network with a single output node. The structure of Σ-Π-Σ is shown in Fig. 2a.

Figure 2b shows a Σ-Π-Σ-Π structure, where the weights between the input layer and Π_1 and between Σ_1 and Π_2 are fixed to 1. The output of Π_1, which is also the input to Σ_1, is determined solely by the input vector. Thus, we can ignore the original input layer and take Π_1 as the input layer. In this sense, Σ-Π-Σ-Π is equivalent to Σ-Π-Σ as far as the learning procedure and the convergence analysis are concerned.

In a Σ-Π-Σ-Π, if Π_2 contains the same number of nodes as Σ_1, and the value of each node of Π_2 copies the value of the corresponding node of Σ_1 (i.e. the connection between Π_2 and Σ_1 is a one-to-one connection), then such a Σ-Π-Σ-Π becomes a Σ-Σ-Π. Hence, Σ-Σ-Π shown in Fig. 2c is a special case of Σ-Π-Σ-Π.

To sum up, in this paper, we shall concentrate our attention to Σ-Π-Σ, and the convergence results are also valid for Σ-Π-Σ-Π and Σ-Σ-Π. The key point here is that our convergence analysis allows any kind of connection (cf. Λ_i and V_j for a Sigma-Pi Unit) between Π-Σ.

Note that the output of Π in a Σ-Σ-Π, which is also the input to Σ_1, is determined solely by the input layer since the weights between Π and the input layer are fixed. Thus, one can even show that a Σ-Σ (cf. Fig. 2d), which is actually the ordinary feed-forward neural networks with a hidden layer, is equivalent to Σ-Σ-Π as far as the learning procedure and the convergence analysis are concerned.

2.3 Σ-Π-Σ Neural Networks

Let us describe the working procedure of a Σ-Π-Σ (cf. Fig. 2a). M, N and Q stand for the numbers of the nodes of the input layer, the Σ_1 layer and the Π layer respectively. We denote the weight vector connecting Π and Σ_2 by $w_0 = (w_{0,1}, \ldots, w_{0,Q})^T \in \mathbb{R}^Q$, and the weight matrix connecting the input layer and Σ_1 by $\widetilde{W} = (w_1, \ldots, w_N)^T \in \mathbb{R}^{N\times M}$, where $w_n = (w_{n1}, \ldots, w_{nM})^T$ $(1 \le n \le N)$ is the weight vector connecting the input layer and the n-th node of Σ_1. Set $W = (w_0^T, w_1^T, \ldots, w_N^T) \in \mathbb{R}^{Q+NM}$. The weights connecting Π and Σ_1 are fixed to 1.

Assume that $g : \mathbb{R} \to \mathbb{R}$ is a given sigmoid activation function which squashes the outputs of the summation nodes. For any $z = (z_1, \ldots, z_N)^T \in \mathbb{R}^N$, we define

$$G(z) = (g(z_1), g(z_2), \ldots, g(z_N))^T. \tag{2}$$

Let $\xi \in \mathbb{R}^M$ be an input vector. Then the output vector ζ of Σ_1 is computed by

$$\zeta = G(\widetilde{W}\xi) = (g(w_1 \cdot \xi), g(w_2 \cdot \xi), \ldots, g(w_N \cdot \xi))^T. \tag{3}$$

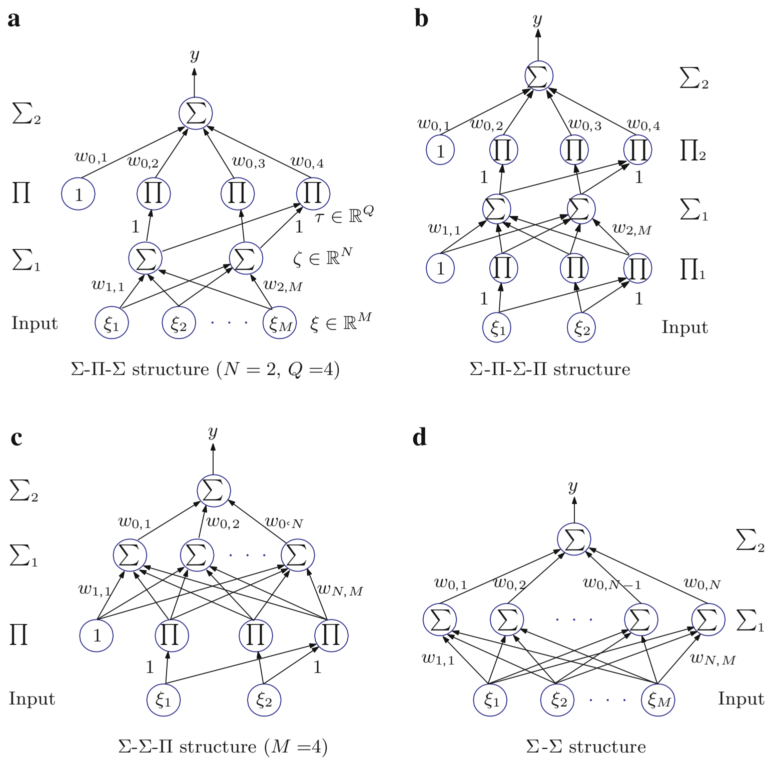

Fig. 2 Four classes of network structures

Denote the output vector of Π by $\tau = (\tau_1, \ldots, \tau_Q)^T$. The component τ_q $(1 \leq q \leq Q)$ is a partial product of the components of the vector ζ. As before, we denote by Λ_q $(1 \leq q \leq Q)$ the index set composed of the indexes of vector ζ 's components connected with τ_q. Then, the output τ_q is computed by

$$\tau_q = \prod_{\lambda \in \Lambda_q} \zeta_\lambda, \quad 1 \leq q \leq Q. \tag{4}$$

The final output of the Σ-Π-Σ network is

$$y = g(w_0 \cdot \tau). \tag{5}$$

2.4 Batch Gradient Learning Algorithm for Σ-Π-Σ

Let the network be supplied with a given set of learning samples $\{\xi^j, O^j\}_{j=1}^J \subset \mathbb{R}^M \times \mathbb{R}$. Let $y^j \in \mathbb{R}$ $(1 \leq j \leq J)$ be the output for each input $\xi^j \in \mathbb{R}^M$. The usual square error function is as follows:

$$E(W) = \frac{1}{2}\sum_{j=1}^{J}(y^j - O^j)^2 \equiv \sum_{j=1}^{J} g_j(w_0 \cdot \tau^j), \tag{6}$$

where

$$g_j(t) = \frac{1}{2}\left(g(t) - O^j\right)^2, \quad t \in \mathbb{R},\ 1 \le j \le J, \tag{7}$$

$$\tau^j = (\tau_1^j, \tau_2^j, \ldots, \tau_Q^j)^T = \left(\prod_{\lambda\in\Lambda_1} \zeta_\lambda^j, \prod_{\lambda\in\Lambda_2} \zeta_\lambda^j, \ldots, \prod_{\lambda\in\Lambda_Q} \zeta_\lambda^j\right)^T, \tag{8}$$

$$\begin{aligned}\zeta^j &= (\zeta_1^j, \zeta_2^j, \ldots, \zeta_N^j) = G(\widetilde{W}\xi^j)\\ &= \left(g(w_1\cdot\xi^j), g(w_2\cdot\xi^j), \ldots, g(w_N\cdot\xi^j)\right)^T.\end{aligned} \tag{9}$$

Then, the partial gradient of the error function $E(W)$ with respect to w_0 is

$$E_{w_0}(W) = \sum_{j=1}^J g_j'(w_0\cdot\tau^j)\tau^j. \tag{10}$$

Moreover, for any $1 \le n \le N$ and $1 \le q \le Q$,

$$\frac{d\tau_q}{dw_n} = \left(\prod_{\lambda\in\Lambda_q\backslash\{n\}} \zeta_\lambda\right) g'(w_n\cdot\xi)\xi, \quad \text{if } n \in \Lambda_q; \tag{11}$$

and if $n \notin \Lambda_q$, $\frac{d\tau_q}{dw_n} = 0$.

$$E_{w_n}(W) = \sum_{j=1}^J g_j'(w_0\cdot\tau^j)\left(\sum_{q=1}^Q w_{0,q}\frac{d\tau_q^j}{dw_n}\right),\ 1 \le n \le N, \tag{12}$$

where $\frac{d\tau_q^j}{dw_n}$ denotes the value of $\frac{d\tau_q}{dw_n}$ at $\zeta_\lambda = \zeta_\lambda^j$ and $\xi = \xi^j$ in (11). According to (4), (11) and (12), for any $1 \le n \le N$, we have

$$E_{w_n}(W) = \sum_{j=1}^J g_j'(w_0\cdot\tau^j)\left(\sum_{q\in V_n} w_{0,q}\left(\prod_{\lambda\in\Lambda_q\backslash\{n\}} \zeta_\lambda^j\right) g'(w_n\cdot\xi^j)\xi^j\right), \tag{13}$$

where V_n is the index set composed of the indexes of vector τ^j 's components connected with ζ_n.

The purpose of the network learning is to find W^* such that

$$E\left(W^*\right) = \min E\left(W\right). \tag{14}$$

A common simple method to solve this problem is the gradient algorithm. Starting from an arbitrary initial values W^0, we proceed to refine the the weights after each cycle of learning iteration. There are two ways of adapting the weights, updating the weights after presentation of each input vector or a batch of input vectors, referred to as online or batch versions, respectively. This paper adheres to the batch version. So in the iteration process, we refine the weights as follows:

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

where $\Delta W^k = (\Delta w_0^k, \Delta w_1^k, \ldots, \Delta w_N^k)$,

$$\Delta w_0^k = -\eta E_{w_0}(W) = -\eta \sum_{j=1}^{J} g_j'(w_0^k \cdot \tau^j)\tau^j, \quad k = 0, 1, 2, \ldots, \tag{16}$$

and, according to (13), for any $1 \leq n \leq N$ and $k = 0, 1, 2, \ldots,$

$$\begin{aligned}\Delta w_n^k &= -\eta E_{w_n}(W) \\ &= -\eta \sum_{j=1}^{J} g_j'(w_0^k \cdot \tau^j) \left(\sum_{q \in V_n} w_{0,q}^k \left(\prod_{\lambda \in \Lambda_q \backslash \{n\}} \zeta_\lambda^j \right) g'(w_n^k \cdot \xi^j)\xi^j \right).\end{aligned} \tag{17}$$

$\eta > 0$ here stands for the learning rate.

3 Main Results

A set of assumptions (A) are first specified:

(A1) $|g(t)|$, $|g'(t)|$ and $|g''(t)|$ are uniformly bounded for any $t \in \mathbb{R}$;
(A2) $\|w_0^k\|_{k=0}^{\infty}$ are uniformly bounded;
(A3) The learning rate η is small enough such that (47) below is valid;
(A4) There exists a bounded set D such that $\{W^k\}_{k=0}^{\infty} \subset D$, and the set $D_0 = \{W \in D : E_W(W) = 0\}$ contains finite points.

If Assumptions $(A1)$–$(A2)$ are valid, we can find a constant $C > 0$ such that

$$\max_{t \in \mathbb{R}, k \in \mathbb{N}} \left\{ \|w_0^k\|, |g(t)|, |g'(t)|, |g''(t)| \right\} \leq C. \tag{18}$$

In the sequel, we will use C for a generic positive constant, which may be different in different places.

Now we are in a position to present the main theorems.

Theorem 1 *Let the error function $E(W)$ be defined in (6), and the sequence $\{W^k\}$ be generated by the Σ-Π-Σ neural network (15)–(17) with W^0 being an arbitrary initial guess. If Assumptions $(A1)$–$(A3)$ are valid, then we have*
(i) $E(W^{k+1}) \leq E(W^k), \quad k = 0, 1, 2, \ldots;$
(ii) $\lim_{k\to\infty} \left\| E_{w_n}(W^k) \right\| = 0, \quad 0 \leq n \leq N;$
Furthermore, if Assumption $(A4)$ also holds, there exists a point $W^* \in D_0$ such that
(iii) $\lim_{k\to\infty} W^k = W^*$.

Theorem 2 *The same conclusions as in Theorem 1 are valid for Σ-Π-Σ-Π, Σ-Σ-Π and Σ-Σ neural networks.*

4 Appendix

In this appendix, we first present two lemmas, then we use them to prove the main theorems.

Lemma 1 *Suppose that $f : R^Q \longrightarrow R$ is continuous and differentiable on a compact set $\tilde{D} \subset R^Q$, and that $\Omega = \{z \in \tilde{D} | \nabla f(z) = 0\}$ has only finite number of points. If a sequence*

$\{z^k\}_{k=1}^{\infty} \subset \tilde{D}$ *satisfies*

$$\lim_{k\to\infty} \|z^{k+1} - z^k\| = 0, \ \lim_{k\to\infty} \|\nabla f(z^k)\| = 0,$$

then there exists a point $z^* \in \Omega$ *such that* $\lim_{k\to\infty} z^k = z^*$.

Proof This result is almost the same as Theorem 14.1.5 in [13] (cf. [14]), and the detail of the proof is omitted. □

For any $k = 0, 1, 2, \ldots, 1 \le j \le J$ and $1 \le n \le N$, we define the following notations.

$$\tau^{k,j} = \tau(\widetilde{W}^k \xi^j), \quad \psi^{k,j} = \tau^{k+1,j} - \tau^{k,j}, \quad \phi_0^{k,j} = w_0^k \cdot \tau^{k,j}, \quad \phi_n^{k,j} = w_n^k \cdot \xi^j. \quad (19)$$

Lemma 2 *Suppose Assumptions* $(A1)$–$(A2)$ *hold, then we have*

$$|g_j'(t)| \le C, \quad |g_j''(t)| \le C, \quad t \in \mathbb{R}; \quad (20)$$

$$\|\psi^{k,j}\|^2 \le C \sum_{n=1}^{N} \|\Delta w_n^k\|^2, \quad 1 \le j \le J, \ k = 0, 1, 2 \ldots; \quad (21)$$

$$\sum_{j=1}^{J} g_j'(\phi_0^{k,j})(w_0^k \cdot \psi^{k,j}) \le -\sum_{n=1}^{N} \eta \|E_{w_n}(W^k)\|^2 + C\eta^2 \sum_{n=1}^{N} \|E_{w_n}(W^k)\|^2; \quad (22)$$

$$\sum_{j=1}^{J} g_j'(\phi_0^{k,j})(\tau^{k,j} \cdot \Delta w_0^k) = -\eta \|E_{w_0}(W^k)\|^2; \quad (23)$$

$$\sum_{j=1}^{J} g_j'(\phi_0^{k,j})(\Delta w_0^k \cdot \psi^{k,j}) \le C\eta^2 \sum_{n=0}^{N} \|E_{w_n}(W^k)\|^2; \quad (24)$$

$$\frac{1}{2} \sum_{j=1}^{J} g_j''(s_{k,j})(\phi_0^{k+1,j} - \phi_0^{k,j})^2 \le C\eta^2 \sum_{n=0}^{N} \|E_{w_n}(W^k)\|^2. \quad (25)$$

where C *is independent of* k, *and* $s_{k,j} \in \mathbb{R}$ *lies on the segment between* $\phi_0^{k,j}$ *and* $\phi_0^{k+1,j}$.

Proof By (7),

$$g_j'(t) = g'(t)(g(t) - O^j),$$
$$g_j''(t) = g''(t)(g(t) - O^j) + (g'(t))^2, \quad 1 \le j \le J, \ t \in \mathbb{R}.$$

Then, (20) follows directly from Assumption $(A1)$.

In order to prove (21), we need the following identity, which can be shown by an induction argument.

$$\prod_{n=1}^{N} a_n - \prod_{n=1}^{N} b_n = \sum_{n=1}^{N} \left(\prod_{s=1}^{n-1} a_s\right)\left(\prod_{t=n+1}^{N} b_t\right)(a_n - b_n), \quad (26)$$

where we have made the convention that $\prod_{s=1}^{0} a_s \equiv 1$ and $\prod_{t=N+1}^{N} b_t \equiv 1$. By (19), (8), (9) and (26), we have for any $1 \le q \le Q$ that

$$\begin{aligned}\psi_q^{k,j} &= \tau_q^{k+1,j} - \tau_q^{k,j} = \prod_{n\in\Lambda_q} g(\phi_n^{k+1,j}) - \prod_{n\in\Lambda_q} g(\phi_n^{k,j}) \\ &= \sum_{n\in\Lambda_q}\left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j})\right)\left(g(\phi_n^{k+1,j}) - g(\phi_n^{k,j})\right), \qquad (27)\end{aligned}$$

where $\Lambda'_{q,n} = \{r | r < n, r \in \Lambda_q\}$ and $\Lambda''_{q,n} = \{r | r > n, r \in \Lambda_q\}$. Here we have made the convention that

$$\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j}) \equiv 1; \quad \prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j}) \equiv 1,$$

when $\Lambda'_{q,n} = \emptyset$ and $\Lambda''_{q,n} = \emptyset$, respectively.

It follows from (27), Assumption $(A1)$, the Mean Value Theorem and the Cauchy–Schwartz Inequality that for any $1 \le j \le J$ and $k = 0, 1, 2, \ldots,$

$$\begin{aligned}\left\|\psi^{k,j}\right\|^2 &\le C\left\|\left(\sum_{n\in\Lambda_1}\left|g(\phi_n^{k+1,j}) - g(\phi_n^{k,j})\right|, \ldots, \sum_{n\in\Lambda_Q}\left|g(\phi_n^{k+1,j}) - g(\phi_n^{k,j})\right|\right)^T\right\|^2 \\ &= C\left\|\left(\sum_{n\in\Lambda_1}\left|g'(t_{k,j,n})(\Delta w_n^k \cdot \xi^j)\right|, \ldots, \sum_{n\in\Lambda_Q}\left|g'(t_{k,j,n})(\Delta w_n^k \cdot \xi^j)\right|\right)^T\right\|^2 \\ &= C\sum_{q=1}^{Q}\left(\sum_{n\in\Lambda_q}\left|g'(t_{k,j,n})(\Delta w_n^k \cdot \xi^j)\right|\right)^2 \\ &\le C\sum_{n=1}^{N}\|\Delta w_n^k\|^2, \qquad (28)\end{aligned}$$

where $t_{k,j,n}$ is on the segment between $\phi_n^{k+1,j}$ and $\phi_n^{k,j}$. This proves (21).

Next, we prove (22). Using the Taylor expansion and (19), we have

$$g(\phi_n^{k+1,j}) - g(\phi_n^{k,j}) = g'(\phi_n^{k,j})(\Delta w_n^k \cdot \xi^j) + \frac{1}{2} g''(\tilde{t}_{k,j,n})(\Delta w_n^k \cdot \xi^j)^2, \qquad (29)$$

where $\tilde{t}_{k,j,n}$ is on the segment between $\phi_n^{k+1,j}$ and $\phi_n^{k,j}$. According to (27), we have

$$w_0^k \cdot \psi^{k,j} = \sum_{q=1}^{Q} w_{0,q}^k \sum_{n\in\Lambda_q}\left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j})\right)\left(g(\phi_n^{k+1,j}) - g(\phi_n^{k,j})\right). \qquad (30)$$

The combination of (29) and (30) leads to

$$\sum_{j=1}^{J} g'_j(\phi_0^{k,j})(w_0^k \cdot \psi^{k,j}) = \delta_1 + \delta_2, \qquad (31)$$

where

$$\delta_1 = \sum_{j=1}^{J} g_j'(\phi_0^{k,j}) \sum_{q=1}^{Q} w_{0,q}^k \sum_{n\in\Lambda_q} \left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j})\right) \times g'(\phi_n^{k,j})(\xi^j \cdot \Delta w_n^k), \tag{32}$$

$$\delta_2 = \frac{1}{2}\sum_{j=1}^{J} g_j'(\phi_0^{k,j}) \sum_{q=1}^{Q} w_{0,q}^k \sum_{n\in\Lambda_q} \left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j})\right) \times g''(\tilde{t}_{k,j,n})(\xi^j \cdot \Delta w_n^k)^2, \tag{33}$$

and $\Lambda'_{q,n} = \{r | r < n, r \in \Lambda_q\}$, $\Lambda''_{q,n} = \{r | r > n, r \in \Lambda_q\}$. For any $1 \le q \le Q$ and $n \in \Lambda_q$, we define

$$\pi_1(q,n) = \left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j})\right) g'(\phi_n^{k,j})(\xi^j \cdot \Delta w_n^k), \tag{34}$$

$$\begin{aligned}\pi_2(q,n) &= \left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k,j})\right) g'(\phi_n^{k,j})(\xi^j \cdot \Delta w_n^k) \\ &= \left(\prod_{\lambda\in\Lambda_q\setminus\{n\}} \zeta_\lambda^j\right) g'(w_n^k \cdot \xi^j)(\xi^j \cdot \Delta w_n^k).\end{aligned} \tag{35}$$

Let us re-write (32) as

$$\delta_1 = \sum_{j=1}^{J} g_j'(\phi_0^{k,j}) \sum_{q=1}^{Q} w_{0,q}^k \sum_{n\in\Lambda_q} \left(\pi_2(q,n) + (\pi_1(q,n) - \pi_2(q,n))\right). \tag{36}$$

According to (1), (13) and (17), we can get

$$\begin{aligned}&\sum_{j=1}^{J} g_j'(\phi_0^{k,j}) \sum_{q=1}^{Q} w_{0,q}^k \sum_{n\in\Lambda_q} \pi_2(q,n) \\ &\quad= \sum_{j=1}^{J} g_j'(\phi_0^{k,j}) \sum_{n=1}^{N} \left(\sum_{q\in V_n} w_{0,q}^k \pi_2(q,n)\right) \\ &\quad= \sum_{n=1}^{N} E_{w_n}(W^k) \cdot \Delta w_n^k = -\eta \sum_{n=1}^{N} \|E_{w_n}(W^k)\|^2.\end{aligned} \tag{37}$$

Then using (26) and the Mean Value Theorem, we have

$$\begin{aligned}
&\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j}) - \prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k,j}) \\
&\quad = \sum_{\lambda\in\Lambda''_{q,n}} \left(\prod_{s\in\Upsilon'_{q,n,\lambda}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Upsilon''_{q,n,\lambda}} g(\phi_t^{k+1,j})\right)\left(g(\phi_\lambda^{k+1,j}) - g(\phi_\lambda^{k,j})\right) \\
&\quad = \sum_{\lambda\in\Lambda''_{q,n}} \left(\prod_{s\in\Upsilon'_{q,n,\lambda}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Upsilon''_{q,n,\lambda}} g(\phi_t^{k+1,j})\right) g'(t_{k,j,\lambda})(\xi^j\cdot\Delta w_\lambda^k), \qquad (38)
\end{aligned}$$

where $t_{k,j,\lambda}$ is on the segment between $\phi_\lambda^{k+1,j}$ and $\phi_\lambda^{k,j}$, $\Upsilon'_{q,n,\lambda} = \{r|r < \lambda, r\in\Lambda''_{q,n}\}$, and $\Upsilon''_{q,n,\lambda} = \{r|r > \lambda, r\in\Lambda''_{q,n}\}$. By (34), (35), (38) and (18), we have the following estimate:

$$\begin{aligned}
&|\pi_2(q,n) - \pi_1(q,n)| \\
&\quad = \left|\left(\prod_{s\in\Lambda'_{q,n}} g(\phi_s^{k,j})\right)\left(\prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k+1,j}) - \prod_{t\in\Lambda''_{q,n}} g(\phi_t^{k,j})\right) g'(\phi_n^{k,j})(\xi^j\cdot\Delta w_n^k)\right| \\
&\quad \le C\left(\sum_{\lambda\in\Lambda''_{q,n}} \|\Delta w_\lambda^k\|\right)\|\Delta w_n^k\|, \qquad (39)
\end{aligned}$$

where $1\le q\le Q$ and $n\in\Lambda_q$. In terms of (1), (18), (20), (38) and (39), we have

$$\begin{aligned}
&\sum_{j=1}^{J} g'_j(\phi_0^{k,j}) \sum_{q=1}^{Q} w_{0,q}^k \sum_{n\in\Lambda_q} (\pi_1(q,n) - \pi_2(q,n)) \\
&\quad = \sum_{j=1}^{J} g'_j(\phi_0^{k,j}) \sum_{n=1}^{N}\sum_{q\in V_n} w_{0,q}^k\,(\pi_1(q,n) - \pi_2(q,n)) \\
&\quad \le C\sum_{n=1}^{N}\sum_{q\in V_n}\left(\left(\sum_{\lambda\in\Lambda''_{q,n}} \|\Delta w_\lambda^k\|\right)\|\Delta w_n^k\|\right) \\
&\quad = C\left(\sum_{n=1}^{N}\|\Delta w_n^k\|\right)\left(\sum_{n=1}^{N}\|\Delta w_n^k\|\right) \le C\sum_{n=1}^{N}\|\Delta w_n^k\|^2. \qquad (40)
\end{aligned}$$

It follows from (36), (37) and (40) that

$$\delta_1 \le -\eta\sum_{n=1}^{N}\|E_{w_n}(W^k)\|^2 + C\eta^2\sum_{n=1}^{N}\|E_{w_n}(W^k)\|^2. \qquad (41)$$

Employing (33), (18) and (17), we obtain

$$\delta_2 \le C\sum_{n=1}^{N}\|\Delta w_n{}^k\|^2 = C\eta^2\sum_{n=1}^{N}\|E_{w_n}(W^k)\|^2. \qquad (42)$$

Now, (22) results from (31), (41), and (42).

(23) is a direct consequence of (10) and (16).

Using (18), (21), (16) and (17), we can show (24) as follows:

$$\sum_{j=1}^{J} g_j'(\phi_0^{k,j})(\Delta w_0^k \cdot \psi^{k,j}) \le C \sum_{j=1}^{J} \|\Delta w_0^k\| \|\psi^{k,j}\|$$
$$\le C \sum_{j=1}^{J} (\|\Delta w_0^k\|^2 + \|\psi^{k,j}\|^2) \le C\eta^2 \sum_{n=0}^{N} \|E_{w_n}(W^k)\|^2. \quad (43)$$

Similarly, a combination of (18), (19), (21), (16) and (17) leads to

$$\frac{1}{2}\sum_{j=1}^{J} g_j''(s_{k,j})(\phi_0^{k+1,j} - \phi_0^{k,j})^2 \le C \sum_{j=1}^{J} |\phi_0^{k+1,j} - \phi_0^{k,j}|^2$$
$$= C \sum_{j=1}^{J} |(w_0^{k+1} - w_0^k) \cdot \tau^{k+1,j} + w_0^k \cdot (\tau^{k+1,j} - \tau^{w,j})|^2$$
$$\le C \sum_{j=1}^{J} \left(\|\Delta w_0^k\| + \|\psi^{k,j}\|\right)^2 \le C\eta^2 \sum_{n=0}^{N} \|E_{w_n}(W^k)\|^2. \quad (44)$$

This proves (25) and completes the proof. □

Now we are ready to prove the main theorems in terms of the above two lemmas.

Proof to Theorem 1 We firstly consider the proof to (i). Using the Taylor expansion, (19), (23), (22) and (25), we have

$$E(W^{k+1}) - E(W^k) = \sum_{j=1}^{J} \left(g_j(\phi_0^{k+1,j}) - g_j(\phi_0^{k,j})\right)$$
$$= \sum_{j=1}^{J} \left(g_j'(\phi_0^{k,j})(\phi_0^{k+1,j} - \phi_0^{k,j}) + \frac{1}{2} g_j''(s_{k,j})(\phi_0^{k+1,j} - \phi_0^{k,j})^2\right)$$
$$= \sum_{j=1}^{J} g_j'(\phi_0^{k,j}) \left(\tau^{k,j} \cdot \Delta w_0^k + w_0^k \cdot \psi^{k,j} + \Delta w_0^k \cdot \psi^{k,j}\right)$$
$$+ \frac{1}{2} \sum_{j=1}^{J} g_j''(s_{k,j})(\phi_0^{k+1,j} - \phi_0^{k,j})^2$$
$$\le -\eta \|E_{w_0}(W^k)\|^2 - \eta \sum_{n=1}^{N} \|E_{w_n}(W^k)\|^2 + C\eta^2 \sum_{n=0}^{N} \|E_{w_n}(W^k)\|^2$$
$$= -(\eta - C\eta^2) \sum_{n=0}^{N} \|E_{w_n}(W^k)\|^2, \quad (45)$$

where $s_{k,j} \in \mathbb{R}$ lies on the segment between $\phi_0^{k,j}$ and $\phi_0^{k+1,j}$. Let $\beta = \eta - C\eta^2$, then

$$E(W^{k+1}) \le E(W^k) - \beta \sum_{n=0}^{N} \left\|E_{w_n}(W^k)\right\|^2. \quad (46)$$

We require the learning rate η to satisfy (C is the constant in (45))

$$0 < \eta < \frac{1}{C}. \tag{47}$$

This together with (46) leads to

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

Next, we prove (ii). By (46), we can get

$$\begin{aligned} E(W^{k+1}) &\leq E(W^k) - \beta \sum_{n=0}^{N} \left\| E_{w_n}(Wk) \right\|^2 \\ &\leq \cdots \leq E(w^0, V^0) - \beta \sum_{t=0}^{k} \left(\sum_{n=0}^{N} \left\| E_{w_n}(W^t) \right\|^2 \right). \end{aligned}$$

Since $E(W^{k+1}) \geq 0$, we have

$$\beta \sum_{t=0}^{k} \left(\sum_{n=0}^{N} \left\| E_{w_n}(W^t) \right\|^2 \right) \leq E(W^0).$$

Letting $k \to \infty$ results in

$$\sum_{t=0}^{\infty} \left(\sum_{n=0}^{N} \left\| E_{w_n}(W^t) \right\|^2 \right) \leq E(W^0) < \infty.$$

So

$$\sum_{k=0}^{\infty} \left\| E_{w_n}(W^k) \right\|^2 \leq \sum_{k=0}^{\infty} \left(\left\| \sum_{n=0}^{N} \right\| E_{w_n}(W^k) \right\|^2 \right) < \infty.$$

This immediately gives

$$\lim_{k \to \infty} \left\| E_{w_n}(W^k) \right\| = 0, \quad 0 \leq n \leq N.$$

Finally, we prove (iii). It follows from (16), (17) and (ii) of Theorem 1 that

$$\lim_{k \to \infty} \|\Delta w_n^k\| = 0, \quad 0 \leq n \leq N. \tag{48}$$

Note that the error function $E(W)$ defined in (6) is continuously differentiable. Using (48), Assumptions $(A3)$–$(A4)$ and Lemma 1, we immediately get the desired result. This completes the proof. □

Proof to Theorem 2 Note that Σ-Π-Σ-Π is equivalent to Σ-Π-Σ by taking Π_1 in Σ-Π-Σ-Π as the input layer as explained in Subsect. 2.2. So Theorem 1 applies to Σ-Π-Σ-Π. Similarly, Theorem 1 applies to Σ-Σ-Π which is a special case of Σ-Π-Σ-Π, and in turn applies to Σ-Σ which is a special case of Σ-Σ-Π. This completes the proof. □

Acknowledgements Wei Wu's work was partly supported by the National Natural Science Foundation of China (10471017).

References

1. Rumelhart DE, McClelland JL (1986) Parallel distributed processing, explorations in the microstructure of cognition. MIT Press, Cambridge
2. Li JY, Yu YL (1995) The realization of arbitrary Boolean function by two layer higher-order neural network. J South China Univ Techn 23:111–116
3. Lenze B (2004) Note on interpolation on the hypercube by means of sigma-pi neural networks. Neurocomputing 61:471–478
4. Bertsekas DP, Tsiklis J (1996) Neuro-dynamic programming. Athena Scientific, Boston, MA
5. Kushner HJ, Yang J (1995) Analysis of adaptive step size SA algorithms for parameter rates. IEEE Transac Automat Control 40:1403–1410
6. Shao HM, Wu W, Li F, Zheng GF (2004) Convergence of gradient algorithm for feedforward neural network training. Proceed Int Symposium Comput Inform 2:627–631
7. Wu W, Feng G, Li X (2002) Training multilayer perceptrons via minimization of sum of ridge functions. Adv Computat Math 17:331–347
8. Liang YC et al (2002) Successive approximation training algorithm for feedforward neural networks. Neurocomputing 42:311–322
9. Wu W, Feng G, Li Z, Xu Y (2005) Deterministic convergence of an online gradient method for BP neural networks. IEEE Transac Neural Networks 16:533–540
10. Durbin R, Rumelhart D (1989) Product units: a computationally powerful and biologically pausible extension to backpropagation networks. Neural Comput 1:133–142
11. Lenze B (1994) How to make sigma-pi neural networks perform perfectly on regular training sets. Neural Networks 7:1285–1293
12. Heywood M, Noakes P (1995) A Framework for improved training of sigma-pi networks. IEEE Transac Neural Networks 6:893–903
13. Yuan Y, Sun W (2001) Optimization theory and methods. Science Press, Beijing
14. Wu W, Shao HM, Qu D (2005) Strong convergence for gradient methods for BP networks training. Proceedings of the International Conference on on Neural Networks and Brains (ICNNB'05), pp 332–334