Neurocomputing 000 (2017) 1–7

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Brief Papers

Relaxed conditions for convergence analysis of online back-propagation algorithm with L_2 regularizer for Sigma-Pi-Sigma neural network☆

Yan Liu[a,b,*], Dakun Yang[c], Chao Zhang[d]

[a] School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116034, China
[b] National Engineering Research Center of Seafood, Dalian 116034, China
[c] School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China
[d] School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China

ARTICLE INFO

Article history:
Received 8 September 2015
Revised 26 July 2016
Accepted 26 June 2017
Available online xxx

Communicated by Bo Shen

Keywords:
L_2 regularizer
Sigma-Pi-Sigma network
Convergence
Boundedness

ABSTRACT

The properties of a boundedness estimations are investigated during the training of online back-propagation method with L_2 regularizer for Sigma-Pi-Sigma neural network. This brief presents a unified convergence analysis, exploiting theorems of White for the method of stochastic approximation. We apply the method of regularizer to derive estimation bounds for Sigma-Pi-Sigma network, and also give conditions for determinating convergence ensuring that the back-propagation estimator converges almost surely to a parameter value which locally minimizes the expected squared error loss. Besides, some weight boundedness estimations are derived through the squared regularizer, after that the boundedness is exploited to prove the convergence of the algorithm. A simulation is also given to verify the theoretical findings.

1. Introduction

Sigma-Pi-Sigma ($\Sigma - \Pi - \Sigma$) neural network (SPSNN) is a kind of high-order network which can learn to implement static mapping in a resemblant manner to that of multilayer neural networks [1]. Due to its output possesses of product-of-sum type, SPSNN is expected to own stronger modeling capability and overcome difficulties of mapping in high-dimensional input space, which employs rich internal nonlinear dynamics, making them suitable for dynamic system identification. Literatures illustrate that the new structure conquers difficulties in function approximation and high-dimensional mapping, which might be encountered when using multilayer neural networks and radial basis function networks.

However, when training the networks with the sum-squared error function, the weights iterated through online gradient method sometimes become giant and the phenomenon of over-fitting is easy to appear. A standard technique to prevent over-fitting is L_2 regularizer added to the original loss error, which is a term proportional to the magnitude of the weights for punishing large weights [2]. Regularization is an elegant and trustworthy trick of obtaining better generalization through neural network training. The fundamental principles of regularizer method are outlined for both linear and nonlinear functions and then extended to overlap some hybrid training algorithm for feedforward neural networks. The idea of functional regularizer is also recommended and studied in high-order networks [3].

Online gradient method has been a widely used and popular learning algorithm for networks. There has been some concentration on the convergence analyses of the learning methods for feedforward networks. For example, the deterministic convergence for the networks was deliberated in [4]. Zhang et al. [5] discussed the online gradient method with penalty term, in which the patterns are presented in a stochastic ordered sequence. [6] settled down the convergence analysis issue with any analytic sigmoid activation function. On the other hand, [7] discussed the convergence of online BP training algorithm under the condition that when the

☆ This work is supported by the National Natural Science Foundation of China (Nos. 61403056, 61473328 and 11401076), Natural Science Foundation Guidance Project of Liaoning Province, China Postdoctoral Science Foundation (no. 2015M581334) and Natural Science Foundation Guidance Project of Liaoning Province (no. 201602050).

* Corresponding author at: School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116034, China.
E-mail address: liuyan@dlpu.edu.cn (Y. Liu).

http://dx.doi.org/10.1016/j.neucom.2017.06.057
0925-2312/

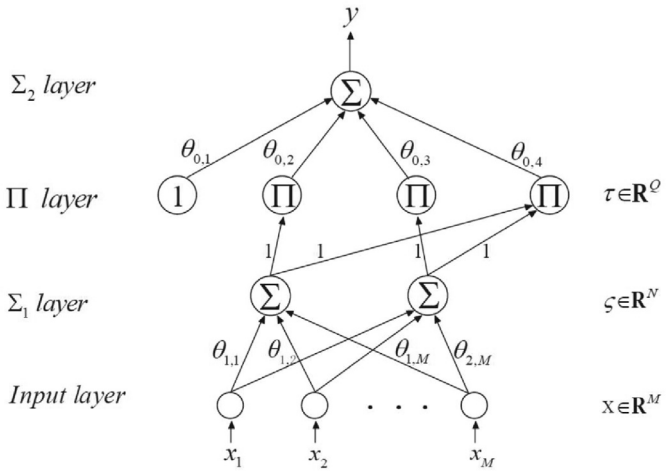

Fig. 1. A Sigma-Pi-Sigma network.

activation function of the neural networks is linear. [8] researched the convergence of the learning method by exploiting the stochastic approximation theory in [9]. Other convergence results obtained through probabilistic asymptotic analysis and deterministic analysis could be referred to [10–17].

But there remains an absence of theoretical guarantee on the Sigma-Pi-Sigma network, especially for online format. One main objective of this paper is to accomplish this gap by proving that for Sigma-Pi-Sigma network the weights in the training procedure are indeed bounded for the online gradient method with L_2 regularizer. Moreover, the techniques demonstrated in the convergence proof could be generalized to other online cases and used to gain the similar convergence conclusions if this regularizer is also added to the original lost error. Specifically, we make the following contributions:

(1) Online BP algorithm with L_2 regularizer is utilized to train SPSNN, in which L_2 regularizer acts as a brute-force to prevent the magnitude of the weights from getting too large in the training process.

(2) The boundedness of the network weights is one crucial condition for the convergence analysis. However, these boundedness conditions might be hard to check. Owing to L_2 regularizer, we prove the weights are automatically bounded.

(3) Furthermore, the convergence of online BP algorithm with L_2 regularizer for SPSNN is proved, in which the boundedness properties are derived and hence the boundedness is no longer needed as a precondition otherwise than pre-mentioned literatures ([13,14]).

The rest of this brief is organized as follows. Section 2 provides a brief introduction to the Sigma-Pi-Sigma network and its online BP method with L_2 regularizer. The main results are presented in Section 3. The rigorous proofs of the main results are carried out in Section 4. Simulation results are given in Section 5. Finally, Section 6 provide a conclusion.

2. Sigma-Pi-Sigma network and its online BP method with L_2 regularizer

2.1. Sigma-Pi-Sigma ($\Sigma - \Pi - \Sigma$) network

The structure of a SPSNN is composed of different high-order neurons. Fig. 1 illustrates the arrangement of Sigma-Pi-Sigma network, which consist of an input layer, two hidden layers of the summing layer (Σ_1 layer) and the product layer (Π layer), and an output layer (Σ_2 layer). The number of neurons of these layers are M, N, Q and 1, respectively. Let $\{d^j, \mathbf{x}^j\}_{j=1}^J \in \mathbb{R} \times \mathbb{R}^M$ be the given training example set, where d^j is the corresponding desired output for $\mathbf{x}^j$. Denote by $\boldsymbol{\theta}_0 = (\theta_{0,1}, \ldots, \theta_{0,Q})^T \in \mathbb{R}^Q$ the weight vector linking the Σ_2 layer and the Π layer. The weights linking the Π layer and the Σ_1 layer are fixed to constant 1. The weight vector linking the input layer and the nth neuron of the Σ_1 layer is $\boldsymbol{\theta}_n = (\theta_{n,1}, \ldots, \theta_{n,M})^T \in \mathbb{R}^M$ $(1 \le n \le N)$. Here, we remark that $x_M^j \equiv -1$ $(j = 1, \ldots, J)$, then θ_{nM} $(n = 1, \ldots, N)$ are corresponding thresholds for the input layer. This disposal blends these thresholds into the weight vectors for unified notation. We write $\boldsymbol{\chi} = (d, \mathbf{x}^T)^T$ and all the weights into one vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_0^T, \boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T)^T \in \mathbb{R}^{Q+NM}$. In the SPSNN, the linear activation function is usually exploited for the product units, while the sigmoidal functions $\phi, \psi : \mathbb{R} \to \mathbb{R}$ are the activation functions for the Σ_1 layer and the Σ_2 layer, respectively.

The flow of the SPSNN goes as follows. Given an input pattern $\mathbf{x} \in \mathbb{R}^M$, after which the output of Σ_1 layer is $\boldsymbol{\zeta} \in \mathbb{R}^N$ could be computed by

$$\boldsymbol{\zeta} = \Big(\zeta_1, \zeta_2, \ldots, \zeta_N\Big)^T = \Big(\phi(\boldsymbol{\theta}_1 \cdot \mathbf{x}), \phi(\boldsymbol{\theta}_2 \cdot \mathbf{x}), \ldots, \phi(\boldsymbol{\theta}_N \cdot \mathbf{x})\Big)^T, \quad (1)$$

where "$\cdot$" is the inner product. For any $\mathbf{a} = (a_1, \ldots, a_N)^T \in \mathbb{R}^N$, a vector function is defined by

$$\Phi(\mathbf{a}) = \Big(\phi(a_1), \phi(a_2), \ldots, \phi(a_N)\Big)^T. \quad (2)$$

Then, the output vector for the Σ_1 layer is

$$\boldsymbol{\zeta} = \Phi(\widetilde{\boldsymbol{\theta}}\mathbf{x}) = \Big(\phi(\boldsymbol{\theta}_1 \cdot \mathbf{x}), \phi(\boldsymbol{\theta}_2 \cdot \mathbf{x}), \ldots, \phi(\boldsymbol{\theta}_N \cdot \mathbf{x})\Big)^T,$$

where $\widetilde{\boldsymbol{\theta}} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N)^T \in \mathbb{R}^{N \times M}$.

As in Fig. 1, each product node of the Π layer is connected with certain nodes (say {1, 2}, {1}, or {2}) of the Σ_1 layer and is in accordance with a particular polynomial (say, correspondingly, x_1x_2, x_1 or x_2). The N nodes in the Σ_1 layer and the Π layer could be fully connected as shown in Fig. 1 with $N = 2$. At this moment, the number of the Π layer is $C_N^0 + C_N^1 + C_N^2 + \cdots + C_N^N = 2^N$. The Σ_1 layer and the Π layer are sparsely connected when the number of the Π layer is less than 2^N. This polynomial, that is, the output of the Π layer, is actually constructed by a weighted linear incorporation such as $\theta_{0,1} + \theta_{0,2}x_1 + \theta_{0,3}x_3 + \theta_{0,4}x_1x_2$.

The output of the Π layer is $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_Q)^T \in \mathbb{R}^Q$ after $\boldsymbol{\zeta}$, in which the component τ_q $(1 \le q \le Q)$ is a part product of $\boldsymbol{\zeta}$. Specifically, let Λ_q $(1 \le q \le Q)$ be the subscript set in which the vector $\boldsymbol{\zeta}$'s components linked to τ_q. Then, τ_q is calculated by

$$\tau_q = \prod_{\lambda \in \Lambda_q} \zeta_\lambda, \quad 1 \le q \le Q. \quad (3)$$

The final output of $\Sigma - \Pi - \Sigma$ network, i.e. the output of the Σ_2 layer is

$$y = \psi(\boldsymbol{\theta}_0 \cdot \boldsymbol{\tau}). \quad (4)$$

2.2. The online BP method with L_2 regularizer for SPSNN

Recall that given an input pattern $\mathbf{x}$, its desired output is d, and the actual output of the network is y, then the instant loss function is given according to the following equation,

$$\hat{\mathcal{E}}(\boldsymbol{\theta}, \boldsymbol{\chi}) = \frac{1}{2}(d - y)^2 = \frac{1}{2}\Big(d - \psi(\boldsymbol{\theta}_0 \cdot \boldsymbol{\tau})\Big). \quad (5)$$

A common practice to add a regularizer term to the error function during training, which is sum of the squared weights. Hence, the complete error function to be minimized during the training process is

$$\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\chi}) = \hat{\mathcal{E}}(\boldsymbol{\theta}, \boldsymbol{\chi}) + \frac{1}{2}\lambda\|\boldsymbol{\theta}\|^2$$

Please cite this article as: Y. Liu et al., Relaxed conditions for convergence analysis of online back-propagation algorithm with L_2 regularizer for Sigma-Pi-Sigma neural network, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.06.057