IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 9, 2020, accepted December 24, 2020, date of publication December 30, 2020,
date of current version February 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048235

Convergence of a Gradient-Based Learning Algorithm With Penalty for Ridge Polynomial Neural Networks

QINWEI FAN[1,2], JIGEN PENG[1], HAIYANG LI[1], AND SHOUJIN LIN[1]
[1]School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006, China
[2]School of Science, Xi'an Polytechnic University, Xi'an 710048, China

Corresponding author: Jigen Peng (jgpeng@gzhu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 11771347, and in part by the 65th China Postdoctoral Science Foundation under Grant 2019M652837.

ABSTRACT Recently there have been renewed interests in high order neural networks (HONNs) for its powerful mapping capability. Ridge polynomial neural network (RPNN) is an important kind of HONNs, which always occupies a key position as an efficient instrument in the tasks of classification or regression. In order to make the convergence speed faster and the network generalization ability stronger, we introduce a regularization model for RPNN with Group Lasso penalty, which deals with the structural sparse problem at the group level in this paper. Nevertheless, there are two main obstacles for introducing the Group Lasso penalty, one is numerical oscillation and the other is convergence analysis challenge. In doing so, we adopt smoothing function to approximate the Group Lasso penalty to overcome these drawbacks. Meanwhile, strong and weak convergence theorems, and monotonicity theorems are provided for this novel algorithm. We also demonstrate the efficiency of our proposed algorithm by numerical experiments, and compare it to the no regularizer, L_2 regularizer, $L_{1/2}$ regularizer, smoothing $L_{1/2}$ regularizer, and the Group Lasso regularizer, and also the relevant theoretical analysis has been verified.

INDEX TERMS Convergence, high order neural networks, ridge polynomial neural network, smoothing Group Lasso.

I. INTRODUCTION

Traditional neural networks have nonlinear mapping capability, which can approximate reasonable functions with arbitrary precision. In recent years, they have been used in various diversified areas. However, the performance of this network is tremendously affected by its parameters. Especially when the training sample dimension is relatively high, the training process of the common neural network architecture becomes extremely slow and tedious [1].

In order to overcome this time-consuming operation and reduce the complexity of the network. Recently, HONNs has been playing an increasingly important role in diverse fields due to its fast convergence speed, strong approximation ability, strong fault tolerance, and large storage capacity [2], [3]. Moreover, HONNs has been developing rapidly in terms of basic theory, model and algorithm, implementation and application, and various HONNs models have also been proposed [4]–[7].

Pi-sigma neural network (PSNN) as a special genre of HONNs was developed by Shin and Ghosh [4], which with addition of some higher order logical processing units in the earlier feedforward neural networks (FNNs). This network has a powerful function approximation capability and also avoids the combination blow-up of higher order terms. However, it is not a universal approximation. In [5], the authors proposed another type of HONNs, namely the ridge polynomial neural network (RPNN), which is an expansion of the PSNN and adopts multiple PSNNs act its basic building blocks. Compared with general feedforward neural networks, RPNN not only has higher efficiency but also retains strong nonlinear mapping ability [8]. At present, RPNN has been widely used in many fields, such as time series prediction [9], [10], financial signal forecasting [11], pattern recognition [12], and classifier based on genetic algorithm [13]. However, along with the order of network becomes excessively high,

The associate editor coordinating the review of this manuscript and approving it for publication was Edith C.-H. Ngai.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

they suffer from the combinatorial explosion of required higher order terms. Therefore, it is necessary to optimize the network structure of RPNN to improve its application efficiency.

Gradient method is one of the classical algorithms for neural network training [14]–[19]. It uses training samples to train the connection weights multiple times, updates the weights after each training, and terminates the training when certain conditions are met. In addition, learning efficiency and theoretical analysis are important problems in the study and application of neural networks. For learning efficiency, most of research is mainly committed to using less numbers of connections (weights) and nodes to achieve the same or better accuracy through structural optimization [20]–[24]. In terms of theoretical analysis, it focuses on the convergence and stability analysis of the algorithms [25]–[27]. We know that RPNN has a more efficient and stable structure, but there are relatively few theoretical studies on it at present, so it is very necessary to carry out the research on this related models.

For improving learning efficiency, there is no evidence that the more hidden nodes yield the better generalization. The usual strategy is to find the most suitable network architecture. It can be roughly divided into growing method [28]–[30], which starts from a small initial network, and then increases the number of hidden neurons during the training process. The other is pruning technology, it starts from a large initial network and then prunes it [31]–[33]. The penalty term for pruning FNNs is widely used for weight elimination, which is used to remove the unnecessary connections. Nevertheless, it is particularly important to determine a compact network structure under the premise of ensuring the generalization ability of the network.

For a standard RPNN, the error function is defined as the sum of the squares of the errors

$$\widetilde{E}(W) = \frac{1}{2}\sum(O^a - y^a)^2 \tag{I.1}$$

where O^a is the ideal output, and y^a is the actual output. When we need to prune the network, a typical strategy is to add a penalty term to the standard error function, and we can get the following error function.

$$E(W) = \frac{1}{2}\sum(O^a - y^a)^2 + \lambda\|W\|_p^p \tag{I.2}$$

The parameter $\lambda > 0$ is the regularization coefficient, which balances the relative importance between network complexity and training accuracy. And $\|W\|_p^p$ is the regularization term, $\|\cdot\|$ stands for the Euclidean norm. For any n dimensional vector, $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$, the L_p norm defined by $\|\alpha\|_p = (|\alpha_1|^p + |\alpha_2|^p + \cdots + |\alpha_n|^p)^{\frac{1}{p}}$.

Generally, there are different forms of regularization terms for different p values. For $p = 2$ is the weight decay regularization term [34]–[36], the increase of network weights can be effectively controlled by weight decay, but this regularization term does not produce a sparse solution. The L_0 regularization term is usually used for variable selection and will produce the sparsest solutions, but these sparse solutions are difficult to calculate. For this reason, L_1 regularization term is proposed, but its sparsity is weaker than L_0 regularization term. In order to make it produce a sparse solution and also easy to calculate, the $L_{1/2}$ regularization term is proposed. Even the L_0, L_1, and $L_{1/2}$ regularizers can generate sparse results, but only the single sparse weight can be selected. So, it is still a challenge for us to decide which neuron is redundant [25], [37]–[40]. As a meaningful extension of Lasso, Group Lasso mainly has to do with variable selection on groups. It is one of the most useful tools for the problem of variable selection by shrinking some estimates of parameters exactly toward zero [41]–[43]. In addition, practical gradient learning algorithms with Group Lasso regularization and the relevant theoretical analysis are still lacking in the literature.

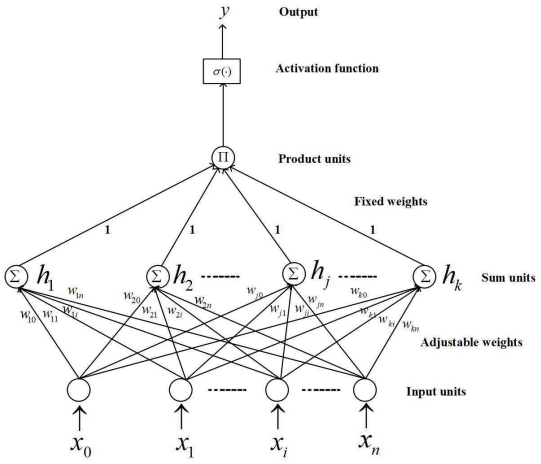


FIGURE 1. Structure of the Pi-Sigma neural network.

In this study, to improve the sparsity of neural networks and enhance generalization ability, we will rely on the idea of group sparsity to optimize the architecture of RPNN (more details in the next section). Toward this end, we would like to satisfy both sparsity of groups and within each group, and the gradient descent method is used to minimize the error function. However, due to the usual Group Lasso regularization term is not differentiable at the origin, and the oscillations of the gradient training process caused. Therefore, we propose a novel smoothing algorithm to overcome these mathematical challenges. The main contributions are as follows:

(i) In order to obtain sparse RPNN with gradient training algorithm, a special smoothing technique is used to approximate the Group Lasso regularization in a small neighborhood near the origin of coordinates. In this way, the novel smoothing Group Lasso regularization, which not only overcomes undifferentiability but also retains the well sparse property.

(ii) This novel pruning algorithm based on Group Lasso method for RPNN, it penalizes the weights at group level by reducing the weight vectors to zero, which more efficient compared with other various pruning strategies.

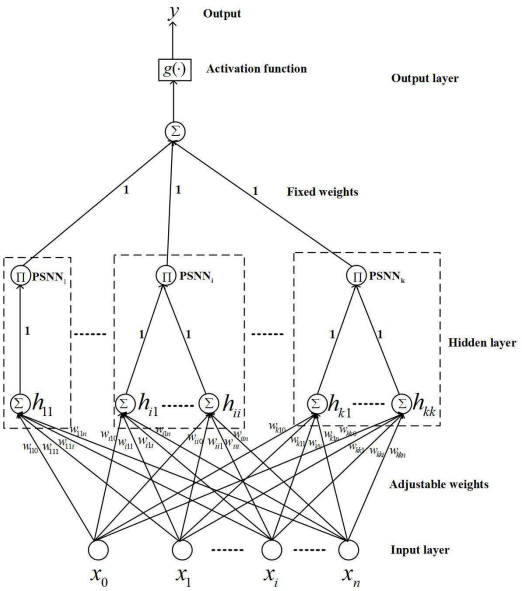


FIGURE 2. Structure of the ridge polynomial neural network.

(iii) Under certain assumptions, we give the strong and weak convergence theorems and monotonicity theorems for the novel algorithm. Moreover, the theoretical results and the advantages of this algorithm are also illustrated by numerical experiments.

The remainder of this article is organized as follows. The preliminaries for the structure of PSNN and RPNN are depicted in the section II. In Section III, we show more details for gradient method with Group Lasso regularization for RPNN. Two theorems of monotonicity, weak and strong convergence are presented in Section IV. Numerical experimental results are shown in Section V to support our theoretical results. We conclude this paper with some useful remarks in Section VI. The detailed proofs of the theoretical results are provided in Appendix.

II. NETWORK STRUCTURE DESCRIPTION

A. PI-SIGMA NEURAL NETWORK (PSNN)

As a kind of more powerful nonlinear mapping network, ridge polynomial neural network (RPNN) is mainly composed of some Pi-Sigma neural network (PSNN). So, we first introduce the structure of PSNN. In Fig.1, it shows the topological structure of PSNN with a single output which is a kth order neural network.

The number of input units, sum units and output unit of the PSNN are $n+1$, k, and 1, respectively. We use $w_j = (w_{j0}, w_{j1}, \ldots, w_{jn}) \in \mathbb{R}^{n+1}$ to denote the weight vectors from the sum unit j to the input units. For the convenience of later calculation, we note $w = (w_1, w_2, \ldots, w_k) \in \mathbb{R}^{(n+1)k}$, and in order to reduce the training convergence time of the network, the weight vectors connecting the sum units and product unit are fixed as 1. Also, let $x = (x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1}$ be $n+1$ dimensional input vector, and the adjustable threshold $x_0 = 1$.

And y express the output of the product unit as follows:

$$y = \sigma(\Pi_{j=1}^{k}(\sum_{i=0}^{n} w_{ji}x_i)) \triangleq \sigma(\Pi_{j=1}^{k}\langle w_j, x\rangle) \qquad \text{(II.1)}$$

where the weights from the sum unit j to the input unit i is denoted by w_{ji}, $\langle\cdot,\cdot\rangle$ and $\sigma(.,.)$ denote the inner product and sigmoid activation function respectively.

B. RIDGE POLYNOMIAL NEURAL NETWORK (RPNN)

Fig.2 shows the topological structure of RPNN, which consists of k PSNNs, and we call k is the order of the network. $PSNN_i$ is the ith PSNN with i summing units. Also, we noticed that it is different from the traditional feedforward neural network, only the weights between the hidden layer and the input layer of RPNN can adjust. The weight vectors from the sum unit j in $PSNN_i$ to the input units is denoted by $w_{ij} = (w_{ij0}, w_{ij1}, \ldots, w_{ijn}) \in \mathbb{R}^{n+1}$. For the sake of simplicity, we rewrite all the weights into a compact set $w = (w_{11}, w_{21}, w_{22}, \cdots, w_{i1}, w_{i2}, \cdots w_{ii}, \cdots, w_{k1}, w_{k2}, \cdots w_{kk}) \in \mathbb{R}^{\frac{k(k+1)}{2}(n+1)}$.

For any multivariate continuous function defined on the compact set of multidimensional input space, RPNN can be used to approximate it with arbitrary precision. A Ridge polynomial is a polynomial represented by $\langle\cdot,\cdot\rangle$, and any multivariate polynomial can be indicated in the form of a ridge polynomial, and implemented by the RPNN, whose output y is expressed as follows:

$$y = g(\sum_{i=1}^{k} P_i(w_i, x)) = g(\sum_{i=1}^{k}\prod_{j=1}^{i}\langle w_{ij}, x\rangle) \qquad \text{(II.2)}$$

where $p_i(w_i, x)$ is the output of the PSNN, and the $g(\cdot)$ is sigmoid activation function.

III. BATCH GRADIENT LEARNING ALGORITHM FOR RPNN

In this section, let the training samples be $(x^a, O^a)_{a=1}^{A}$, where x^a and O^a are the input and desired output respectively. And let y^a be the actual output for each input x^a.

A. THE ORIGINAL GROUP LASSO REGULARIZATION ALGORITHM

For the batch gradient learning algorithm for RPNN with Group Lasso regularizer, let $E(w)$ be the mean squared error function with Group Lasso penalty term. Then, we can get the following formula

$$\begin{aligned} E(w) &= \frac{1}{2}\sum_{a=1}^{A}(O^a - y^a)^2 + \lambda\sum_{i=1}^{k}\sum_{j=1}^{i}\|w_{ij}\| \\ &= \sum_{a=1}^{A}\frac{1}{2}(O^a - g(\sum_{i=1}^{k}P_i(x^a)))^2 + \lambda\sum_{i=1}^{k}\sum_{j=1}^{i}\|w_{ij}\| \end{aligned} \qquad \text{(III.1)}$$

For the sake of brevity, denoted by

$$f_a(\sum_{i=1}^{k} P_i(x^a)) \triangleq \frac{1}{2}(O^a - g(\sum_{i=1}^{k} P_i(x^a)))^2 \quad \text{(III.2)}$$

$$P_i(x^a) \triangleq P_i(w_i, x^a) \triangleq \prod_{j=1}^{i} \langle w_{ij}, x^a \rangle \quad \text{(III.3)}$$

$$\varphi_{a,k} \triangleq \sum_{i=1}^{k} P_i(x^a) \quad \text{(III.4)}$$

Then, the following equivalent form can be obtained

$$\begin{aligned} E(w) &= \sum_{a=1}^{A} f_a(\sum_{i=1}^{k} P_i(x^a)) + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i} \|w_{ij}\| \\ &= \sum_{a=1}^{A} f_a(\sum_{i=1}^{k} \prod_{j=1}^{i} \langle w_{ij}, x^a \rangle) + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i} \|w_{ij}\| \end{aligned} \quad \text{(III.5)}$$

the partial derivative $\nabla_{w_{ij}} E(w)$ of $E(w)$ with respect to w_{ij} is

$$\begin{aligned} \nabla_{w_{ij}} E(w) &\triangleq E_{w_{ij}}(w) \\ &= \sum_{a=1}^{A} f_a'(\sum_{i=1}^{k} P_i(x^a)) \prod_{s=1, s\neq j}^{i} \langle w_{is}, x^a \rangle \cdot x^a + \lambda \frac{w_{ij}}{\|w_{ij}\|} \end{aligned} \quad \text{(III.6)}$$

So,

$$E_w = (E_{w_{11}}, E_{w_{21}}, E_{w_{22}}, \cdots, E_{w_{k1}}, E_{w_{k2}}, \cdots, E_{w_{kk}}). \quad \text{(III.7)}$$

And then, in the iteration process, the weights updating rule as follows:

$$w^{m+1} = w^m + \triangle w^m, \quad m = 0, 1, 2, \cdots \quad \text{(III.8)}$$

and

$$\begin{aligned} \triangle w_{ij}^m &= -\eta \nabla_{w_{ij}^m} E(w^m) \\ &= -\eta [\sum_{a=1}^{A} f_a'(\sum_{i=1}^{k} P_i(x^a)) \prod_{s=1, s\neq j}^{i} \langle w_{is}^m, x^a \rangle \cdot x^a + \lambda \frac{w_{ij}^m}{\|w_{ij}^m\|}] \end{aligned} \quad \text{(III.9)}$$

where $\eta > 0$ is the learning rate.

Because this is not differentiable at the origin which may be prone to yield oscillation phenomenon, and this is a great challenge for theoretical analysis. So, we proposed the following smoothing Group Lasso regularization algorithm.

B. THE SMOOTHING GROUP LASSO REGULARIZATION ALGORITHM

In order to overcome the previously mentioned nonderivative defects. First, define the following smoothing function $h(z)$

$$h(z) = \begin{cases} \|z\|, & \|z\| > \alpha \\ \dfrac{\|z\|^2}{2\alpha} + \dfrac{\alpha}{2}, & \|z\| \leq \alpha \end{cases} \quad \text{(III.10)}$$

and then, the gradient of with $h(z, \alpha)$ respect to vector z as follows:

$$\nabla_z h(z) = \begin{cases} \dfrac{z}{\|z\|}, & \|z\| > \alpha \\ \dfrac{z}{\alpha}, & \|z\| \leq \alpha \end{cases} \quad \text{(III.11)}$$

Thus, we can get

$$\begin{aligned} \tilde{E}(w) &= \frac{1}{2}\sum_{a=1}^{A}(O^a - y^a)^2 + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i} h(w_{ij}) \\ &= \sum_{a=1}^{A} \frac{1}{2}(O^a - g(\sum_{i=1}^{k} P_i(x^a)))^2 + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i} h(w_{ij}) \\ &= \sum_{a=1}^{A} f_a(\sum_{i=1}^{k} P_i(x^a)) + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i} h(w_{ij}) \\ &= \sum_{a=1}^{A} f_a(\sum_{i=1}^{k} P_i \langle w_i, x^a \rangle) + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i} h(w_{ij}) \end{aligned} \quad \text{(III.12)}$$

where $f_a(\cdot)$ is defined as (III.2).

Then starting with an initial value w^0, the weights updating rule for w is iteratively by

$$w^{m+1} = w^m + \triangle w^m, \quad m = 0, 1, 2, \cdots \quad \text{(III.13)}$$

and

$$\begin{aligned} \triangle w_{ij}^m &= -\eta \nabla_{w_{ij}^m} E(w^m) \\ &= -\eta [\sum_{a=1}^{A} f_a'(\sum_{i=1}^{k} P_i(x^a)) \prod_{s=1, s\neq j}^{i} \langle w_{is}^m, x^a \rangle \cdot x^a \\ &\quad + \lambda \nabla_{w_{ij}^m} h(w_{ij}^m)] \end{aligned} \quad \text{(III.14)}$$

where $\eta > 0$ is the learning rate.

Remark: In the above definition of $h(z)$, we notice that α is a very small positive constant.

IV. THEOREMS OF MONOTONICITY AND CONVERGENCE

Throughout this section, some assumptions are needed to prove the main theorems.

Assumption A1: For any $t \in \mathbb{R}$, the $f_a'(t)$ and $f_a''(t)$ are bounded, i.e., there exists constant $C_1 > 0$ such that

$$\max\{|f_a'(t)|, |f_a''(t)|\} \leq C_1$$

Assumption A2: There exists a constant C_2 such that

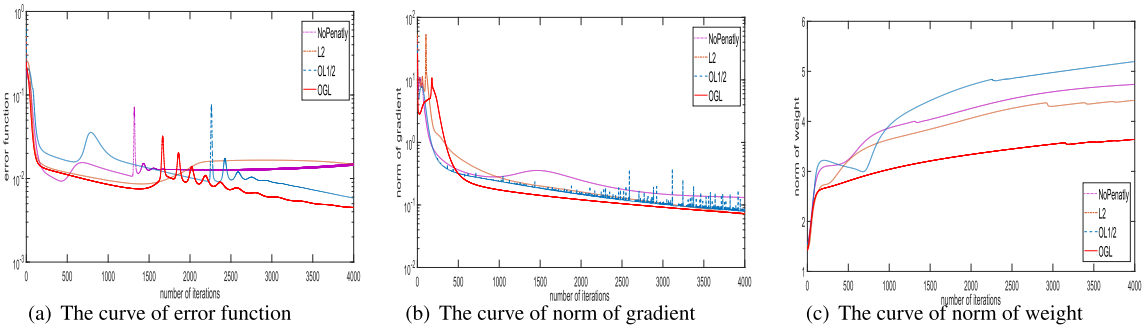
$$\max\{|w_{ij}x^a|, \|x^a\|\} \leq C_2, 1 \leq i \leq k, 1 \leq j \leq i, 1 \leq a \leq A$$

Assumption A3: The learning rate η satisfies the condition as follows:

$$0 < \eta < \frac{4}{2A(k-1)C_1C_2^k + Ak(k+1)C_1C_2^{2k} + 4M\lambda}$$

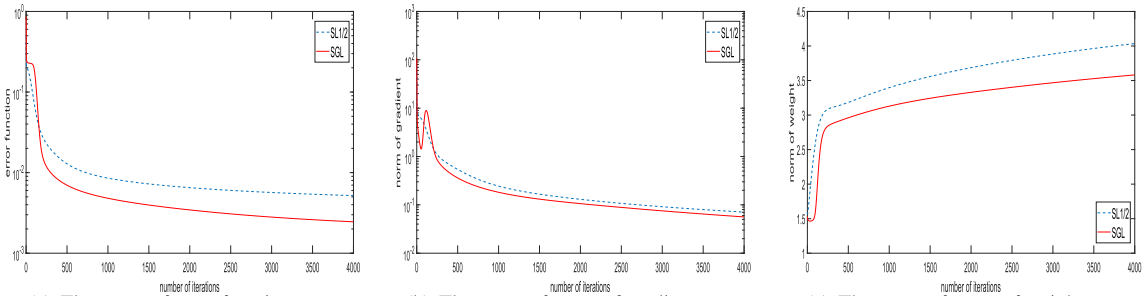
where $M = \frac{\sqrt{6}}{4\sqrt{a^3}}$.

Assumption A4: The set $\bar{\Omega} = \{w \in D | \nabla_w E(w) = 0\}$ contains finite point and the weight sequence $\{w^n\}_{n=0}^{\infty} \subset D$, where D is a compact set.



(a) The curve of error function (b) The curve of norm of gradient (c) The curve of norm of weight

FIGURE 3. The performance result of the *NoPenatly*, L_2, $OL_{1/2}$ and *OGL*.



(a) The curve of error function (b) The curve of norm of gradient (c) The curve of norm of weight

FIGURE 4. The performance result of the $SL_{1/2}$ and *SGL*.

Next, we give the main theoretical results.

Theorem 1: For an arbitrary initial value, the weight sequence $\{w^n\}_{n=0}^{\infty}$ *is generated by* (III.13)*, and the assumptions* $A1 - A3$ *are satisfied, then we have*

(i) $E(w^{n+1}) \leq E(w^n), \quad n = 0, 1, 2, \ldots$

(ii) $\lim_{n\to\infty} E(w^n) = E^* \geq 0.$

Theorem 2: For an arbitrary initial value, the weight sequence $\{w^n\}_{n=0}^{\infty}$ *is generated by* (III.13)*, and the assumptions* $A1 - A4$ *are satisfied, then we have*

(i) $\lim_{n\to\infty} \|\nabla E(w^n)\| = 0.$

(ii) There exists a $w^* \in D_0$ *such that* $\lim_{n\to\infty} (w^n) = w^*.$

V. NUMERICAL EXPERIMENTAL RESULTS

To confirm the validity of the proposed new algorithm based on the smoothing Group Lasso regularization (*SGL*) of the RPNN, we compare it with the no regularizer, L_2 regularizer (L_2), the original $L_{1/2}$ regularizer ($OL_{1/2}$), the smoothing $L_{1/2}$ regularizer ($SL_{1/2}$) and the original Group Lasso regularizer (*OGL*) by using two examples: a nonlinear function approximation and a benchmark problem-parity problem. All experiments of these algorithms are performed in the Matlab 2017a environment running on a Windows personal computer with Intel Core i5-10210U 1.60 2.11 GHz CPUs and RAM 8 GB.

A. EXAMPLE 1: FUNCTION APPROXIMATION PROBLEM

In this example, the following nonlinear function has been thought out to compare the approximation capabilities of the above algorithms:

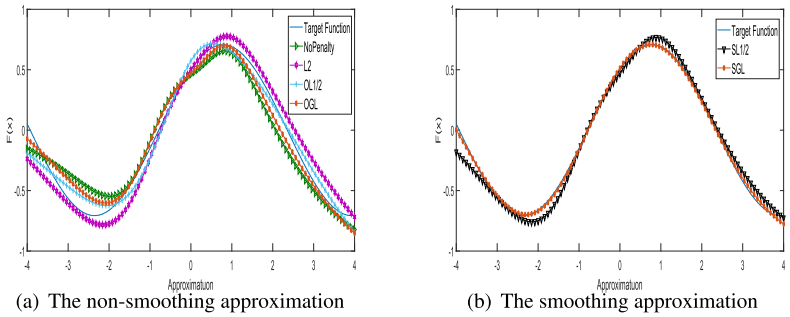
$$F(x) = \frac{1}{2}(sin(x) + cos(x)).$$

In this experiment, the network structure model of RPNN composed of five PSNN modules is adopted, and chooses the sigmoid function as the activation function, and selects 101 training samples from an evenly spaced interval of $[-4, 4]$. The initial weights are chosen randomly within the interval $[-0.5, 0.5]$, and the learning rate η is 0.003, the regularization coefficient λ is 0.002. In regard to each learning algorithm we conducted 10 experiments, and all algorithms are run 4,000 iterations for per experiment. For computational complexity, we need to understand the number of iterations of the algorithm and the number of network layers, and when calculating the number of network layers, since the input layer only transmits data and does not participate in the calculation, it can be ignored. For this we know the time complexity is O(4000), space complexity is O(3).

TABLE 1. Performance comparison on nonlinear function.

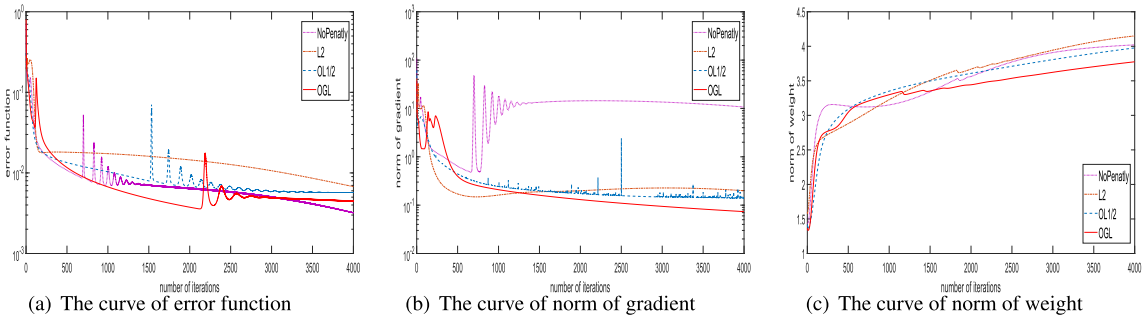
Learning Methods	average error	Time(s)
NoPenalty	0.00645	3.0233
L2	0.00536	2.9005
OL1/2	0.00532	2.8727
SL1/2	0.00427	2.8235
OGL	0.00412	2.8241
SGL	0.00366	2.7348

The average error and running time of 10 experiments are presented in Table 1, and we can see that our proposed



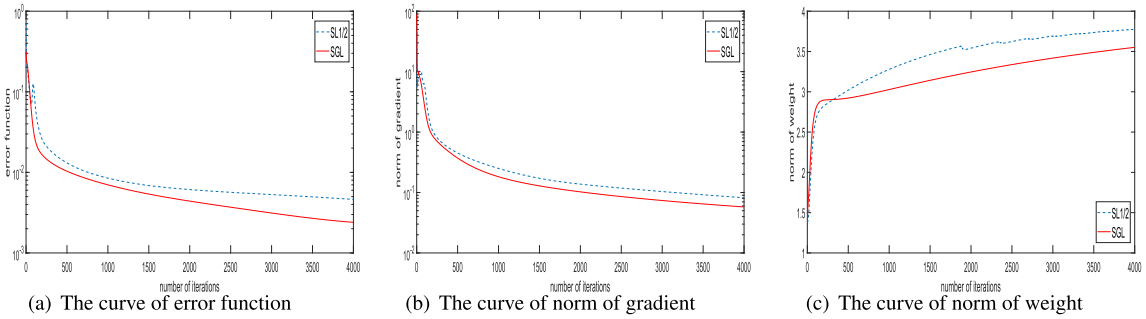
(a) The non-smoothing approximation (b) The smoothing approximation

FIGURE 5. The approximation results of nonlinear functions for all algorithms.



(a) The curve of error function (b) The curve of norm of gradient (c) The curve of norm of weight

FIGURE 6. The performance result based on 5-bit parity problem with ***NoPenatly***, L_2, $OL_{1/2}$ and ***OGL***.



(a) The curve of error function (b) The curve of norm of gradient (c) The curve of norm of weight

FIGURE 7. The performance result based on 5-bit parity problem with $SL_{1/2}$ and ***SGL***.

smoothing Group Lasso regularizer (*SGL*) algorithm for RPNN is surpass the no regularizer, the L_2 regularizer (L_2), the original $L_{1/2}$ regularizer ($OL_{1/2}$), the smoothing $L_{1/2}$ regularizer ($SL_{1/2}$), and the original Group Lasso regularizer (*OGL*). In addition, the performance results of the *NoPenalty*, L_2, $OL_{1/2}$, $SL_{1/2}$, *OGL*, and *SGL* are shown in the Fig.3 and Fig.4. We presented the error function curves, the norm of gradient curves and the curves of the norm of the weights for the no regularizer, L_2, $OL_{1/2}$, and *OGL* algorithms in Fig.3. And similarly, Fig.4 shown the performance results of $SL_{1/2}$, and *SGL*. It is clear that the error function decreases monotonously, and the algorithm we proposed has the smallest error, not only that, it also eliminates the oscillation phenomenon produced by the original algorithm. And from Fig.3(c) and Fig.4(c), we can see that it not only converges faster but also has a better suppression effect in terms of weight growth.

Fig.5 shows the approximation effect of the above algorithms on the nonlinear function, and compared with other algorithms, the proposed new algorithm has better approximation performance.

B. EXAMPLE 2: PARITY PROBLEM

In this experiment, we utilize the solution of the 5-bit parity problem as an example to verify the theoretical effect of our proposed algorithm. Similar to the above approximation experiment, we choose 5 PSNN modules as the basic building blocks of the RPNN structure, and also select 101 training samples from an evenly spaced interval of $[-4, 4]$. In the interval $[-0.5, 0.5]$, the initial weights are randomly selected, and for each learning algorithm, we conducted 10 experiments with the same learning rate η and penalty coefficient λ, which are 0.003, and 0.002, respectively, and all algorithms are run 4,000 iterations for per experiment.

The comparison results of *NoPenatly*, L_2, $OL_{1/2}$,*OGL*, $SL_{1/2}$ and *SGL* algorithms based on the 5-bit parity problem are shown in the Fig.6 and Fig.7. In the figure, we based on the 5-bit parity problem presented the error function curves, the norm of gradient curves and the curve of the norm of the weight for each algorithm, separately. Obviously, we can see that the error function decreases monotonously and eliminates the oscillation phenomenon generated by the original algorithm. In addition, the weight norm curve confirms that the new algorithm we proposed can effectively suppress the weight growth.

VI. CONCLUSION

In this paper, we first propose the Group Lasso regularization learning algorithm for Ridge Polynomial Neural Networks, then a novel pruning model for Ridge Polynomial Neural Networks based on smoothing Group Lasso regularizer is bring forward. By the smoothing technique, we overcome two main drawbacks, one is the difficulty of theoretical analysis and the other is numerical oscillations. More importantly, with some proper assumptions, we get the strong and weak convergence results. Numerical experiments also fully demonstrate that compared with existing regularization learning algorithms, such as the original Group Lasso, L_2 regularization learning algorithm and the $L_{1/2}$ regularization learning algorithm, the smoothing Group Lasso regularization learning algorithm has a better sparse effect. In addition, this regularization method can be used to optimize the traditional BP neural network and fuzzy neural network and extreme learning machine and so on.

APPENDIX

In this section, we will provide a thorough mathematical treatment for Theorem 3 and Theorem 4.

Proof: **Theorem 3 (i):** Applying Taylor mean value theorem with lagrange remainder, we have

$$P_i(w_i^{m+1},x^a)=P_i(w_i^m,x^a)+\sum_{j=1}^{i}\frac{\partial P_i(w_i^m,x^a)}{\partial w_{ij}^m}\cdot\triangle w_{ij}^m+R_{a,i}^m \tag{VII.1}$$

where

$$R_{a,i}^m=\frac{1}{2}(\triangle w_i^m)\nabla^2_{w_iw_i}P_i(t_i^m,x^a)(\triangle w_i^m)^T \tag{VII.2}$$

with $[\nabla^2_{w_iw_i}P_i]_{m,n}=\frac{\partial^2 P_i}{\partial w_{im}\partial w_{in}}$, $t_i^m=(t_{i1}^m,t_{i2}^m,\ldots,t_{ii}^m)$, and t_{ij}^m $(1\le j\le i)$ are between w_{ij}^{m+1} and w_{ij}^m.

That is

$$R_{a,i}^m=\frac{1}{2}(\triangle w_i^m)\begin{pmatrix}\frac{\partial^2 P_i}{\partial w_{i1}\partial w_{i1}} & \frac{\partial^2 P_i}{\partial w_{i1}\partial w_{i2}} & \cdots & \frac{\partial^2 P_i}{\partial w_{i1}\partial w_{i1}}\\ \frac{\partial^2 P_i}{\partial w_{i2}\partial w_{i1}} & \frac{\partial^2 P_i}{\partial w_{i2}\partial w_{i2}} & \cdots & \frac{\partial^2 P_i}{\partial w_{i2}\partial w_{ii}}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial^2 P_i}{\partial w_{ii}\partial w_{i1}} & \frac{\partial^2 P_i}{\partial w_{ii}\partial w_{i2}} & \cdots & \frac{\partial^2 P_i}{\partial w_{ii}\partial w_{ii}}\end{pmatrix}(\triangle w_i^m)^T \tag{VII.3}$$

It is obvious that

$$\frac{\partial^2 P_i}{\partial w_{i1}\partial w_{i1}}=0,\ \frac{\partial^2 P_i}{\partial w_{i2}\partial w_{i2}}=0,\cdots,\frac{\partial^2 P_i}{\partial w_{ii}\partial w_{ii}}=0. \tag{VII.4}$$

Then

$$R_{a,i}^m=\frac{1}{2}\sum_{j=1}^{i}\sum_{s=1,s\neq j}^{i}(\triangle w_{ij}^m\cdot\prod_{q=1,q\neq s,j}^{i}\langle w_{iq}^m,x^a\rangle\cdot\|x^a\|^2\cdot(\triangle w_{is}^m)^T). \tag{VII.5}$$

Thus, we have

$$\begin{aligned}\varphi_{a,k}^{m+1}-\varphi_{a,k}^m&=\sum_{i=1}^{k}(P_i(w_i^{m+1},x^a)-P_i(w_i^m,x^a))\\&=\sum_{i=1}^{k}(\sum_{j=1}^{i}\frac{\partial P_i(w_i^m,x^a)}{\partial w_{ij}^m}\cdot\triangle w_{ij}^m+R_{a,i}^m)\end{aligned} \tag{VII.6}$$

Apply Taylor mean value theorem to $f_a(\varphi_{a,k}^{m+1})$ at $\varphi_{a,k}^m$, we have

$$\begin{aligned}f_a(\varphi_{a,k}^{m+1})=f_a(\varphi_{a,k}^m)+f_a'(\varphi_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)\\+\frac{1}{2}f_a''(t_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)^2\end{aligned} \tag{VII.7}$$

where $t_{a,k}^m$ is between $\varphi_{a,k}^m$ and $\varphi_{a,k}^{m+1}$.

Therefore, from the above equality (VII.7), we have

$$\begin{aligned}&E(w^{m+1})-E(w^m)\\&=\sum_{a=1}^{A}f_a(\varphi_{a,k}^{m+1})+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}h(w_{ij}^{m+1})\\&\quad-\sum_{a=1}^{A}f_a(\varphi_{a,k}^m)+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}h(w_{ij}^m)\\&=\sum_{a=1}^{A}[f_a'(\varphi_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)+\frac{1}{2}f_a''(t_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)^2]\\&\quad+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^m)]\end{aligned} \tag{VII.8}$$

Moreover, by the equality (VII.6), we have

$$\begin{aligned}&E(w^{m+1})-E(w^m)\\&=\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)+\sum_{a=1}^{A}\frac{1}{2}f_a''(t_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)^2\\&\quad+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^m)]\\&=\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)(\sum_{i=1}^{k}\sum_{j=1}^{i}(\frac{\partial P_i(w_i^m,x^a)}{\partial w_{ij}^m}\cdot\triangle w_{ij}^m))\\&\quad+\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)(\sum_{i=1}^{k}R_{a,i}^m)+\sum_{a=1}^{A}\frac{1}{2}f_a''(t_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)^2\end{aligned}$$

$$+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^{m})]$$
$$=\sum_{a=1}^{A}f_a'(\varphi_{a,k}^{m})(\sum_{i=1}^{k}\sum_{j=1}^{i}(\frac{\partial P_i(w_i^m,x^a)}{\partial w_{ij}^m}\cdot\triangle w_{ij}^m))$$
$$+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^{m})]+\alpha_1+\alpha_2 \quad \text{(VII.9)}$$

Denote by $\alpha_1=\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)(\sum_{i=1}^{k}R_{a,i}^m)$, $\alpha_2=\sum_{a=1}^{A}\frac{1}{2}\times f_a''(t_{a,k}^m)(\varphi_{a,k}^{m+1}-\varphi_{a,k}^m)^2$.

Next, we get $\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)(\sum_{i=1}^{k}\sum_{j=1}^{i}(\frac{\partial P_i(w_i^m,x^a)}{\partial w_{ij}^m}\cdot\triangle w_{ij}^m))+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^m)]$, α_1 and α_2 are simplified, respectively.

$$\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)(\sum_{i=1}^{k}\sum_{j=1}^{i}(\frac{\partial P_i(w_i^m,x^a)}{\partial w_{ij}^m}\cdot\triangle w_{ij}^m))$$
$$+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^m)]$$
$$=\sum_{i=1}^{k}\sum_{j=1}^{i}\sum_{a=1}^{A}f_a'(\varphi_{a,k}^m)\prod_{s=1,s\neq j}^{i}\langle w_{is}^m,x^a\rangle\cdot x^a\cdot\triangle w_{ij}^m$$
$$+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^m)]$$
$$=\sum_{i=1}^{k}\sum_{j=1}^{i}(-\frac{1}{\eta}\triangle w_{ij}^m-\lambda\nabla_{w_{ij}^m}h(w_{ij}^m))\cdot\triangle w_{ij}^m$$
$$+\lambda\sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1})-h(w_{ij}^m)]$$
$$\leq-(\frac{1}{\eta}-M\lambda)\|\triangle w^m\|^2 \quad \text{(VII.10)}$$

where $\tilde{w}_{ij}^m$ is between w_{ij}^m and w_{ij}^{m+1}, $\|\nabla_{w_{ij}^m}\nabla_{w_{ij}^m}h(\tilde{w}_{ij}^m))\|\leq M$.

$$|\sum_{i=1}^{k}R_{a,i}^m|$$
$$=\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{i}\sum_{s=1,s\neq j}^{i}((\triangle w_{ij}^m\cdot\prod_{q=1,q\neq s,j}^{i}\langle w_{iq}^m,x^a\rangle\cdot\|x^a\|^2\cdot(\triangle w_{is}^m)^T))$$
$$\leq\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{i}\sum_{s=1,s\neq j}^{i}(\frac{1}{2}(\|\triangle w_{ij}^m\|^2+\|\triangle w_{is}^m\|^2))C_2^{i-2}C_2^2$$
$$=\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{i}\sum_{s=1,s\neq j}^{i}\frac{1}{2}(\|\triangle w_{ij}^m\|^2+\|\triangle w_{is}^m\|^2)C_2^i$$
$$=\frac{1}{2}\sum_{i=1}^{k}\frac{1}{2}[\sum_{j=1}^{i}(i-1)\|\triangle w_{ij}^m\|^2+\sum_{j=1}^{i}\sum_{s=1,s\neq j}^{i}\|\triangle w_{is}^m\|^2]C_2^i$$
$$=\frac{1}{2}\sum_{i=1}^{k}\frac{1}{2}\cdot 2(i-1)\sum_{j=1}^{i}\|\triangle w_{ij}^m\|^2C_2^i$$
$$\leq\frac{1}{2}(k-1)C_2^k\sum_{i=1}^{k}\sum_{j=1}^{i}\|\triangle w_{ij}^m\|^2$$
$$\leq\frac{1}{2}(k-1)C_2^k\sum_{i=1}^{k}\|\triangle w^m\|^2 \quad \text{(VII.11)}$$

Therefore, it follows from Assumption A1, Assumption A2, we have

$$|\alpha_1|=|\sum_{a=1}^{A}(f_a'(\varphi_{a,k}^m)\sum_{i=1}^{k}R_{a,i}^m)|$$
$$\leq\sum_{a=1}^{A}(|f_a'(\varphi_{a,k}^m)|\cdot|\sum_{i=1}^{k}R_{a,i}^m|)$$
$$\leq\sum_{a=1}^{A}C_1\cdot\frac{1}{2}(k-1)C_2^k\sum_{i=1}^{k}\|\triangle w^m\|^2 \quad \text{(VII.12)}$$
$$=\frac{1}{2}A(k-1)C_1C_2^k\sum_{i=1}^{k}\|\triangle w^m\|^2$$
$$|\varphi_{a,k}^{m+1}-\varphi_{a,k}^m|$$
$$=|\sum_{i=1}^{k}(P_i(w_i^{m+1},x^a)-P_i(w_i^m,x^a))|$$
$$=|\sum_{i=1}^{k}\prod_{j=1}^{i}\langle w_{ij}^{m+1},x^a\rangle-\sum_{i=1}^{k}\prod_{j=1}^{i}\langle w_{ij}^m,x^a\rangle|$$
$$\leq\sum_{i=1}^{k}|\prod_{j=1}^{i}\langle w_{ij}^{m+1},x^a\rangle-\prod_{j=1}^{i}\langle w_{ij}^m,x^a\rangle|$$
$$\leq\sum_{i=1}^{k}[|\prod_{j=1}^{i-1}\langle w_{ij}^{m+1},x^a\rangle(\langle w_{ii}^{m+1},x^a\rangle-\langle w_{ii}^m,x^a\rangle)|$$
$$+|\prod_{j=1}^{i-1}\langle w_{ij}^{m+1},x^a\rangle-\prod_{j=1}^{i-1}\langle w_{ij}^m,x^a\rangle||\langle w_{ii}^m,x^a\rangle|]$$
$$\leq\sum_{i=1}^{k}[|\prod_{j=1}^{i-1}\langle w_{ij}^{m+1},x^a\rangle(\langle w_{ii}^{m+1},x^a\rangle-\langle w_{ii}^m,x^a\rangle)|$$
$$+|\prod_{j=1}^{i-2}\langle w_{ij}^{m+1},x^a\rangle(\langle w_{i(i-1)}^{m+1},x^a\rangle-\langle w_{i(i-1)}^m,x^a\rangle)|$$
$$\cdot|\langle w_{ii}^m,x^a\rangle|$$
$$+|\prod_{j=1}^{i-3}\langle w_{ij}^{m+1},x^a\rangle(\langle w_{i(i-2)}^{m+1},x^a\rangle-\langle w_{i(i-2)}^m,x^a\rangle)|\cdot$$
$$|\langle w_{ii}^m,x^a\rangle|\cdot|\langle w_{i(i-1)}^m,x^a\rangle|+\cdots$$
$$+\langle|w_{i1}^{m+1},x^a\rangle-\langle w_{i1}^m,x^a\rangle|\cdot|\langle w_{ii}^m,x^a\rangle||\langle w_{i(i-1)}^m,x^a\rangle|$$
$$\cdots|\langle w_{i2}^m,x^a\rangle|] \quad \text{(VII.13)}$$

By further calculation, we get

$$\begin{aligned}
&|\varphi_{a,k}^{m+1} - \varphi_{a,k}^{m}| \\
&\le \sum_{i=1}^{k}[|\prod_{j=1}^{i-1}\langle w_{ij}^{m+1}, x^a\rangle\langle \triangle w_{ii}^{m}, x^a\rangle| \\
&\quad + |\prod_{j=1}^{i-2}\langle w_{ij}^{m+1}, x^a\rangle\langle \triangle w_{i(i-1)}^{m}, x^a\rangle| \cdot |\langle w_{ii}^{m}, x^a\rangle| \\
&\quad + \cdots \\
&\quad + \langle|\langle \triangle w_{i1}^{m}, x^a\rangle| \cdot |\langle w_{ii}^{m}, x^a\rangle||\langle w_{i(i-1)}^{m}, x^a\rangle| \cdots |\langle w_{i2}^{m}, x^a\rangle|] \\
&\le \sum_{i=1}^{k}(C_2^i\|\triangle w_{ii}^{m}\| + C_2^i\|\triangle w_{i(i-1)}^{m}\| + \cdots + C_2^i\|\triangle w_{i1}^{m}\|) \\
&\le C_2^k \sum_{i=1}^{k}\sum_{j=1}^{i}\|\triangle w_{ij}^{m}\| \qquad \text{(VII.14)}
\end{aligned}$$

Therefore, it follows from (VII.14), Assumption A1, Assumption A2, we have

$$\begin{aligned}
|\alpha_2| &= |\sum_{a=1}^{A}\frac{1}{2}f_a''(t_{a,k}^{m})(\varphi_{a,k}^{m+1} - \varphi_{a,k}^{m})^2| \\
&\le A \cdot C_1 \cdot |\varphi_{a,k}^{m+1} - \varphi_{a,k}^{m}|^2 \\
&\le A \cdot C_1 \cdot (C_2^k \sum_{i=1}^{k}\sum_{j=1}^{i}\|\triangle w_{ij}^{m}\|)^2 \\
&\le \frac{1}{4}Ak(k+1)C_1 \cdot C_2^{2k} \cdot \|\triangle w^{m}\|^2 \qquad \text{(VII.15)}
\end{aligned}$$

From (VII.9), (VII.10), (VII.12), and (VII.15), we have

$$\begin{aligned}
&E(w^{m+1}) - E(w^{m}) \\
&= \sum_{a=1}^{A} f_a'(\varphi_{a,k}^{m})(\sum_{i=1}^{k}\sum_{j=1}^{i}(\frac{\partial P_i(w_i^{m}, x^a)}{\partial w_{ij}^{m}} \cdot \triangle w_{ij}^{m})) \\
&\quad + \lambda \sum_{i=1}^{k}\sum_{j=1}^{i}[h(w_{ij}^{m+1}) - h(w_{ij}^{m})] + \alpha_1 + \alpha_2 \\
&\le -(\frac{1}{\eta} - M\lambda)\|\triangle w^{m}\|^2 + \frac{1}{2}A(k-1)C_1C_2^k\sum_{i=1}^{k}\|\triangle w^{m}\|^2 \\
&\quad + \frac{1}{4}Ak(k+1)C_1 \cdot C_2^{2k} \cdot \|\triangle w^{m}\|^2 \\
&= -(\frac{1}{\eta} - \frac{1}{2}A(k-1)C_1C_2^k\sum_{i=1}^{k}(-\frac{1}{4})Ak(k+1) \\
&\quad \times C_1 \cdot C_2^{2k} - M\lambda)\|\triangle w^{m}\|^2 \qquad \text{(VII.16)}
\end{aligned}$$

Hence, we get
$E(w^{m+1}) \le E(w^{m})$.
The proof to (i) of Theorem 3 is completed.

Theorem 3 (ii): From the conclusion of (i), we get the sequence $\{E(w^n)\}$ is monotonic decrease.

Also, we know that $E(w^{m}) \ge 0$, *for* $m = 1, 2, 3, \cdots$, that is to say $\{E(w^{m})\}$ is bounded below.

By monotone bound theory, we know that there must have $E^* \ge 0$ such that $\lim_{m\to\infty} E(w^{m}) = E^*$.

So, the proof of Theorem 4.1 is completed. □

Next, so as to get the Theorem 4.2, we need the following Lemma.

Lemma 1 [44]: $D \subset \mathbb{R}^n$ is a compact set, let suppose $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and differentiable on D. And $\bar{\Omega} = \{x \in D|\nabla f(x) = 0\}$ contains only finite number of points. If a sequence $\{x^k\} \subset D$ satisfies

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

then, there exists $x^* \in \bar{\Omega}$ such that $\lim_{k\to\infty} x_k = x^*$.

Proof:

Theorem 4 (i): Let $\beta = \frac{1}{\eta} - \frac{1}{2}A(k-1)C_1C_2^k\sum_{i=1}^{k} - \frac{1}{4}Ak(k+1)C_1 \cdot C_2^{2k} - M\lambda > 0$ by the assumption (*A*3), the inequality (VII.16) is as follows:

$$E(w^{m+1}) - E(w^{m}) \le -\beta\|\triangle w^{m}\|^2 \qquad \text{(VII.17)}$$

From (VII.17), we get

$$E(w^{m+1}) \le E(w^{m}) - \beta\sigma^m \le \cdots \le E(w^0) - \beta\sum_{k=0}^{m}\sigma^k$$

Since $E(w^{m+1}) \ge 0$, we have

$$\beta\sum_{k=0}^{m}\sigma^k \le E(w^0)$$

Let $n \to \infty$, then

$$\sum_{k=0}^{\infty}\sigma^k \le \frac{1}{\beta}E(w^0) < \infty$$

This results in

$$\lim_{n\to\infty}\sigma^n = 0$$

It follows from (VII.17) that

$$\lim_{m\to\infty}\|\triangle w^{m}\| = 0. \qquad \text{(VII.18)}$$

Hence,

$$\lim_{m\to\infty}\|\nabla E(w^{m})\| = 0. \qquad \text{(VII.19)}$$

Theorem 4 (ii): We noticed the (III.12), the error function $E(w)$ is continuous and differentiable. Considering (VII.18) and (VII.19), Assumption (*A*4) and applying the Lemma (1), the desired result can be easily obtained, i.e., there exists a point $w^* \in D_0$ such that

$$\lim_{n\to\infty}(w^n) = w^*$$

This completes the proof of theorem 4. □

REFERENCES

[1] R. Ghazali, A. J. Hussain, and P. Liatsis, "Dynamic ridge polynomial neural network: Forecasting the univariate non-stationary and stationary trading signals," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3765–3776, Apr. 2011.

[2] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-1, no. 4, pp. 364–378, Oct. 1971.

[3] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in a high-order neural network," *Appl. Opt.*, vol. 26, no. 23, pp. 4972–4978, 1987.

[4] Y. Shin and J. Ghosh, "The Pi-sigma network: An efficient higher-order neural networks for pattern classification and function approximation," *IEEE Trans. Neural Netw.*, to be published.

[5] Y. Shin and J. Ghosh, "Ridge polynomial networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 610–622, May 1995.

[6] C. K. Li, "A sigma-Pi-sigma neural network," *Neural Process. Lett.*, vol. 17, no. 1, pp. 1–19, 2003.

[7] B. Lenze, "Note on interpolation on the hypercube by means of sigma–pi neural networks," *Neurocomputing*, vol. 61, pp. 471–478, Oct. 2004.

[8] X. Yu, L. Tang, Q. Chen, and C. Xu, "Monotonicity and convergence of asynchronous update gradient method for ridge polynomial neural network," *Neurocomputing*, vol. 129, pp. 437–444, Apr. 2014.

[9] W. Waheeb, R. Ghazali, and T. Herawan, "Ridge polynomial neural network with error feedback for time series forecasting," *PLoS ONE*, vol. 11, no. 12, Dec. 2016, Art. no. e0167248.

[10] W. Waheeb and R. Ghazali, "Forecasting the Behavior of gas furnace multivariate time series using ridge polynomial based neural network models," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 5, no. 5, pp. 126–133, 2019.

[11] R. Ghazali, A. Jaafar Hussain, N. Mohd Nawi, and B. Mohamad, "Non-stationary and stationary prediction of financial time series using dynamic ridge polynomial neural network," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2359–2367, Jun. 2009.

[12] C. Voutriaridis, Y. S. Boutalis, and B. G. Mertzios, "Ridge polynomial networks in pattern recognition," in *Proc. EC-VIP-MC 4th EURASIP Conf. Focused Video/Image Process. Multimedia Commun.*, Jul. 2003, pp. 519–524.

[13] C.-T. Lin, M. Prasad, and A. Saxena, "An improved polynomial neural network classifier using real-coded genetic algorithm," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 11, pp. 1389–1401, Nov. 2015.

[14] J. Wang, Y. Wen, Z. Ye, L. Jian, and H. Chen, "Convergence analysis of BP neural networks via sparse response regularization," *Appl. Soft Comput.*, vol. 61, pp. 354–363, Dec. 2017.

[15] J. Wang, B. Zhang, Z. Sun, W. Hao, and Q. Sun, "A novel conjugate gradient method with generalized armijo search for efficient training of feedforward neural networks," *Neurocomputing*, vol. 275, pp. 308–316, Jan. 2018.

[16] W. Wu, J. Wang, M. Cheng, and Z. Li, "Convergence analysis of online gradient method for BP neural networks," *Neural Netw.*, vol. 24, no. 1, pp. 91–98, Jan. 2011.

[17] W. Wu, X. Yan, X. D. Kang, and C. Zhang, "Convergence of asynchronous batch gradient method for pi-sigma neural networks," *Discrete Continuous Dyn. Impuls. Syst. A*, vol. 14, no. S1, pp. 95–99, 2007.

[18] C. Zhang, W. Wu, X. H. Chen, and Y. Xiong, "Convergence of BP algorithm for product unit neural networks with exponential weights," *Neurocomputing*, vol. 72, nos. 1–3, pp. 513–520, Dec. 2008.

[19] Y. Liu, Z. Li, D. Yang, K. S. Mohamed, J. Wang, and W. Wu, "Convergence of batch gradient learning algorithm with smoothing $L_{1/2}$ regularization for sigma–Pi–sigma neural networks," *Neurocomputing*, vol. 151, pp. 333–341, Mar. 2015.

[20] R. Law, "Back-propagation learning in improving the accuracy of neural network-based tourism demand forecasting," *Tourism Manage.*, vol. 21, no. 4, pp. 331–340, Aug. 2000.

[21] W. Ding, J. Zhang, and Y. Leung, "Prediction of air pollutant concentration based on sparse response back-propagation training feedforward neural networks," *Environ. Sci. Pollut. Res.*, vol. 23, no. 19, pp. 1–14, 2016.

[22] J. Wu and M. Liu, "Improving generalization performance of artificial neural networks with genetic algorithms," in *Proc. IEEE Int. Conf. Granular Comput.*, Jul. 2005, pp. 288–291.

[23] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 239–242, Jun. 1990.

[24] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *Fiber*, vol. 56, no. 4, pp. 3–7, 2016.

[25] W. Wu, Q. Fan, J. M. Zurada, J. Wang, D. Yang, and Y. Liu, "Batch gradient method with smoothing regularization for training of feedforward neural networks," *Neural Netw.*, vol. 50, pp. 72–78, Feb. 2014.

[26] T. Pan, J. Zhao, W. Wu, and J. Yang, "Learning imbalanced datasets based on SMOTE and Gaussian distribution," *Inf. Sci.*, vol. 512, pp. 1214–1233, Feb. 2020.

[27] Q. Fan, J. M. Zurada, and W. Wu, "Convergence of online gradient method for feedforward neural networks with smoothing $L_{1/2}$ regularization penalty," *Neurocomputing*, vol. 131, pp. 208–216, May 2014.

[28] S. K. Sharma and P. Chandra, "Constructive neural networks: A review," *Int. J. Eng. Sci. Technol.*, vol. 12, no. 2, pp. 7847–7855, 2010.

[29] S. S. Sridhar and M. Ponnavaikko, "Improved adaptive learning algorithm for constructive neural networks," *Int. J. Comput. Elec. Eng.*, vol. 1, no. 3, pp. 30–36, 2011.

[30] R. Parekh, J. Yang, and V. Honavar, "Constructive neural-network learning algorithms for pattern classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 436–451, Mar. 2000.

[31] Y. Xiong, W. Wu, X. Kang, and C. Zhang, "Training pi-sigma network by online gradient algorithm with penalty for small weight update," *Neural Comput.*, vol. 19, no. 12, pp. 3356–3368, Dec. 2007.

[32] N. M. Wagarachchi and A. S. Karunananda, "Optimization of multi-layer artificial neural networks using delta values of hidden layers," in *Proc. IEEE Symp. Comput. Intell., Cognit. Algorithms, Mind, Brain (CCMB)*, Apr. 2013, pp. 80–86.

[33] F. Li, J. M. Zurada, Y. Liu, and W. Wu, "Input layer regularization of multilayer feedforward neural networks," *IEEE Access*, vol. 5, pp. 10979–10985, 2017.

[34] X. Yu and Q. Chen, "Convergence of gradient method with penalty for ridge polynomial neural network," *Neurocomputing*, vol. 97, pp. 405–409, Nov. 2012.

[35] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[36] W. Wu, H. M. Shao, and Z. X. Li, "Convergence of batch BP algorithm with penalty for FNN training," *Neural Inf. Process.*, vol. 4232, pp. 562–569, 2006.

[37] H. S. Zhang, Y. L. Tang, and X. D. Liu, "Batch gradient training method with smoothing L_0 regularization for feedforward neural networks," *Neural Comput. Appl.*, vol. 26, pp. 383–390, 2015.

[38] Q. Fan and T. Liu, "Smoothing L_0 regularization for extreme learning machine," *Math. Problems Eng.*, vol. 2020, pp. 1–10, Jul. 2020.

[39] X. Xie, H. Zhang, J. Wang, Q. Chang, J. Wang, and N. R. Pal, "Learning optimized structure of neural networks by hidden node pruning with L_1 regularization," *IEEE Trans. Cybern.*, vol. 50, no. 3, pp. 1333–1346, Mar. 2020.

[40] Q. W. Fan, L. Niu, and Q. Kang, "Regression and multiclass classification using sparse extreme learning machine via smoothing group $L_1/2$ regularizer," *IEEE Access*, vol. 8, pp. 191482–191494, 2020.

[41] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *J. Comput. Graph. Statist.*, vol. 22, no. 2, pp. 231–245, May 2012.

[42] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Stat. Soc., Ser. B, Stat. Methodol.*, vol. 68, no. 1, pp. 49–67, Feb. 2006.

[43] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *J. Roy. Stat. Soc., Ser. B, Stat. Methodol.*, vol. 70, no. 1, pp. 53–71, Jan. 2008.

[44] Y. X. Yuan and W. Y. Sun, *Optimization Theory and Methods*. Beijing, China: Science Press, 1997.

QINWEI FAN received the M.S. degree in applied mathematics from Tianjin Polytechnic University, China, in 2010, and the Ph.D. degree in computational mathematics from the Dalian University of Technology, Dalian, China, in 2014. From 2012 to 2013, he was financially supported by the China Scholarship Council (CSC) to work as a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Louisville, USA. He is currently an Associate Professor with the School of Science, Xi'an Polytechnic University. His research interests include neural networks, signal processing, and learning theory.

JIGEN PENG received the B.S. degree in mathematics from Jiangxi University, Jiangxi, China, in 1989, and the M.Sc. and Ph.D. degrees in applied mathematics and computing mathematics from Xi'an Jiaotong University, Xi'an, China, in 1992 and 1998, respectively. He is currently a Professor with the School of Mathematics and Information Science, Guangzhou University, Guangzhou, China. His current research interests include nonlinear functional analysis and applications, data set matching theory, machine learning theory, and sparse information processing.

HAIYANG LI received the B.S. degree in mathematics from Yan'an University, Shaanxi, China, in 1996, and the M.Sc. and Ph.D. degrees in fundamental mathematics from Shaanxi Normal University, Xi'an, China, in 2005 and 2008, respectively. He is currently a Professor with the School of Mathematics and Information Science, Guangzhou University, Guangzhou, China. His current research interests include quantum logic theory, machine learning theory, and sparse information processing.

SHOUJIN LIN received the M.S. degree from the School of Science, Xi'an Jiaotong University, Xi'an, China, in 2003. He is currently working with Zhongshan MLTOR Intelligent Equipment Company Ltd., Zhongshan, China. His current research interests include machine learning theory, advanced manufacturing, and sparse information processing.