AIMS Mathematics, 9(3): 5995–6012.
DOI: 10.3934/math.2024293
Received: 22 October 2023
Revised: 26 December 2023
Accepted: 28 December 2023
Published: 01 February 2024

http://www.aimspress.com/journal/Math

Research article

A new Sigma-Pi-Sigma neural network based on L_1 and L_2 regularization and applications

Jianwei Jiao[1] **and Keqin Su**[2,*]

[1] School of Civil Engineering, Zhengzhou Institute of Finance and Economics, Zhengzhou, 450000, China

[2] College of Information and Management Science, Henan Agricultural University, Zhengzhou, 450046, China

* **Correspondence:** Email: keqinsu1980@163.com.

Abstract: As one type of the important higher-order neural networks developed in the last decade, the Sigma-Pi-Sigma neural network has more powerful nonlinear mapping capabilities compared with other popular neural networks. This paper is concerned with a new Sigma-Pi-Sigma neural network based on a L_1 and L_2 regularization batch gradient method, and the numerical experiments for classification and regression problems prove that the proposed algorithm is effective and has better properties comparing with other classical penalization methods. The proposed model combines the sparse solution tendency of L_1 norm and the high benefits in efficiency of the L_2 norm, which can regulate the complexity of a network and prevent overfitting. Also, the numerical oscillation, induced by the non-differentiability of L_1 plus L_2 regularization at the origin, can be eliminated by a smoothing technique to approximate the objective function.

Keywords: a new Sigma-Pi-Sigma neural network; batch gradient method; regularization; convergence
Mathematics Subject Classification: 68T07, 92B20

1. Introduction

The Sigma-Pi-Sigma neural network (SPSNN) is a feed-forward neural network composed of Sigma-Pi units, which can be used to achieve a static mapping of multi-layer neural networks [1–3]. In [4], a new model which can be regarded as a subclass of networks on Sigma-Pi units was considered, and the authors showed the origin of the Kronecker product representation from the classical Sigma-Pi units. In [5], a Sigma-Pi network and a new arithmetic were proposed, by which the output representation self-organizes to form a topographic map, whose main contribution was to solve the

frame of reference transformation issues through unsupervised learning. Furthermore, in [6–8], the approximation, convergence performance, and generalization ability of sparse Sigma-Pi network functions were studied, and it was shown that the new algorithms were more efficient than those in the existing literatures.

It is widely recognized that neural network optimization has emerged as a highly significant research topic in recent times. Study on neural network optimization consists of two main topics: One is weight optimization, that is, for a given network structure, select the appropriate learning method to seek the optimal weight such that the training error and the generalization error are small enough [9, 10]. The second is structural optimization, namely the selection of appropriate activation function, network layer number, connection mode, and so on [11–13]. But, the research on neural network structure optimization is far less rich than that on weight optimization. On the other hand, there is no literature showing that more hidden layer neurons yield better generalization ability.

When it comes to neural networks, the number of neurons is a crucial factor. There are two common methods for determining the size of networks. The first method is the growing method, where the network starts with a smaller size and new hidden neurons are added during the training process [14]. Another method is the pruning method [15–19], which begins with a larger network and eventually removes redundant nodes.

These kind of algorithms separates weight learning and weight training, which is inefficient. There are also many slightly more complex algorithms, which further introduce various mechanisms like particle swarm optimization [20], genetic algorithms [21], eigenvalue analysis [22], statistical analysis, and synthetic minority over-sampling techniques [23,24], so as to enhance the sparsification efficiency. The disadvantage of these algorithms is that the program is complex and the calculation is large.

An appropriately sized network structure is instrumental in enhancing efficiency. Overfitting poses a significant challenge during network training and is particularly problematic for deep neural network learning [25]. Consequently, researchers have extensively explored various forms of sparse regularization techniques, highlighting their indispensability.

Recent years, L_p regularization is diffusely used to solve variable selection and parameter estimation problems in machine learning. This regularization method takes the form

$$E(W) = \widehat{E}(W) + \tau \|W\|_p^p,$$

where $\widehat{E}$ is the normal error function, $\|W\|_p = (\sum_i |W_i|^p)^{1/p}$ denotes the p-norm, τ is the penalty coefficient, and $\|\cdot\|$ represents the euclidean norm. This regularization term is also named the penalty term.

In general, there are several common forms of regularization: weight elimination, weight decay, and approximate smoothing [26–32]. Among them, weight elimination is widely used as the penalty term in pruning feedforward neural networks, mainly to reduce unnecessary connections or optimize the network weights [33]. A little more detailed introduction for different penalty terms are as follows.

For different values of p, the L_2 norm to the standard error function makes it more optimized [34–37]. This practice is called L_2 regularization, and the form is shown as follows:

$$E(W) = \widehat{E}(W) + \tau \|W\|_2^2,$$

where $\|W\|_2^2$ denotes the penalty term, and the L_2 norm solution is popular because of its special relationship with the normal distributions. It can serve as brute force to avoid excessive weights.

But, unfortunately, it is not sparse. This means that, during the training process, the L_2 regularization can not drive unnecessary weights to zero.

The L_0 regularization term is the ordinary method for feature extraction and variable selection. Constrained by the number of coefficients, the L_0 regularizer produces the most sparse solutions, which are difficult to calculate, and it is a combinatory optimization problem [38, 39].

As an alternative to the L_0 regularization term, the increasingly important L_1 regularization term (Lasso) has become popular since it just needs to resolve a quadratic programming problem [40]. In [41], the L_1-norm was combined with the capped L_1-norm to indicate the amount of information collected by the filter and the control regularization. The L_1 regularization penalty function is generally denoted as follows:

$$E(W) = \widehat{E}(W) + \tau\|W\|_1.$$

It was shown in [42, 43] that, although these algorithms can generate an alternate neural network structure, they do not offer a unified neural network framework to solve a class of problems.

In order to explore more appropriate neural network structures to get over the obstacles posed by suboptimal models, we propose a new SPSNN algorithm based on the L_1 penalty and the L_2 penalty to deal with complex and varied tasks within a unified framework to improve the robustness and generality of the model. Based on the L_1 norm, an L_2 norm is presented in SPSNNs to promote the population sparsity effect to select the relevant hidden node population. Therefore, our proposed variant algorithm benefits from ridge regression and the tendency towards sparse solutions of the L_1 penalty, which generates a more suitable neural network structure than using one of the regular terms alone, and the elastic net algorithm can also be used to solve for these hybrid penalties [44].

In this paper, the penalty method is considered in the case of selecting the weights, which not only overcomes the shortcomings of the L_1 and L_2 norms, (both L_1 and L_2 penalties are used in the same minimization problem), but also has good generalization ability and sparsity. Thus, the mathematical expression for this hybrid penalty as an error function can be expressed as follows:

$$E(W) = \widehat{E}(W) + \tau_1\|W\|_1 + \tau_2\|W\|_2^2,$$

where the tuning constants τ_1 and τ_2 are fixed and non-negative. The role of τ_1 provides a choice of variables through a sparse vector, and τ_2 ensures a unique solution and leads to a grouping effect. A penalty term is added that is a convex combination of the L_1 norm $\|W\|_1$ and the L_2 norm $\|W\|_2^2$ of the parameter W.

During the training progress, the usual mixed regularization terms are not differentiable at the origin, which usually give rise to oscillation phenomenon. Therefore, we propose a new smoothing algorithm to get over these difficulties. That is, we can use the smoothing technique of the weight in the neighborhood of the origin of the error function instead of the absolute value of the weight. The main contribution and novelty of this article are as follows:

(1) In this article, in order to obtain the optimal architecture with good generalization performance, we will eliminate the weights in the hidden layer by using the idea of elastic net regularization. It means that, by incorporating the L_1 and L_2 regularization terms, this novel algorithm not only eliminates unnecessary weights but also performs pruning of the network structure in the hidden layers. This pruning reduces the size of the network, leading to an effective optimization of the network structure.

(2) We propose an SPSNN based on elastic net regularized batch gradient methods to obtain an optimal architecture with good generalization performance. By means of smoothing technology, we effectively get over the oscillation phenomenon in the process of network learning.

(3) To test the accuracy and robustness, we apply the proposed method to a large number of regression and classification tasks and compare it with other algorithms that have good sparsity and generalization capabilities.

The rest of the article is arranged as follows: In Section 2 the SPSNN is presented, the batch gradient algorithm for this model is depicted in Section 3, and the novel pruning algorithm based on L_1 plus L_2 penalty term is depicted for more details. Some numerical experiments are provided in Section 4, and finally some conclusions are made.

2. Sigma-Pi-Sigma neural networks

Next, we mainly depict the fabric of SPSNNs, which are composed of an input layer, a hidden layer of summing nodes($\sum_1$), a hidden layer of product nodes($\prod$), and output layer($\sum_2$). P, N, Q, and 1 represent the number of the nodes of the input layer, $\sum_1$ layer, $\prod$ layer, and $\sum_2$ layer, respectively (see Fig. 1)

We can use $W_0 = (w_{01}, w_{02}, \cdots w_{0Q})^T \in \mathbb{R}^Q$ to express the weight vector which connects $\sum_2$ layer and the $\prod$ layer. Let the weight vector connecting the $\prod$ layer and the $\sum_1$ layer be fixed as 1. Meanwhile, using $W_n = (w_{n1}, w_{n2}, \cdots w_{nP})^T$ as the weight vector which connects the input layer and the *nth* node of the $\sum_1$ layer, we can set

$$W = ({W_0}^T, {W_1}^T, \cdots {W_N}^T)^T \in \mathbb{R}^{Q+NP}.$$

Let us use $X = (x_1, x_2, \cdots x_P) \in \mathbb{R}^P$ to describe the input of the networks. Assume that $g : \mathbb{R} \longrightarrow \mathbb{R}$ is a given sigmoid activation function for the $\sum_1$ layer. We denote the output vector $\xi \in \mathbb{R}^N$ of the $\sum_1$ layer as

$$\xi = (g(W_1 \cdot X), g(W_2 \cdot X), \cdots g(W_N \cdot X))^T,$$

where $\cdot$ denotes the inner product between vectors.

Similarly, we take $\delta = (\delta_1, \delta_2, \cdots \delta_Q)^T$ to indicate the output vector of the $\prod$ layer. The nodes of the network are connected in two different ways. One is fully connected between the $\sum_1$ layer and the $\prod$ layer, and the other one is sparsely connected. The difference between them lies in the number of product nodes: the former has 2^N product nodes, and the latter is less than 2^N. We can clearly see the structure of the SPSNNs in Figure 1.

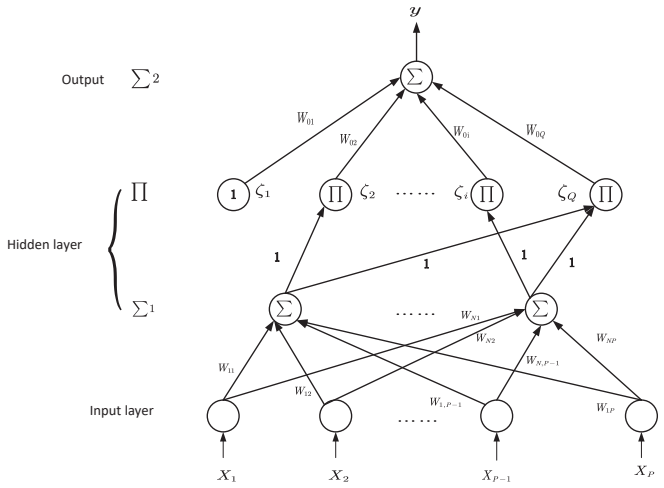


Figure 1. Structure of an SPSNN.

Here, $\mathcal{H}_q(1 \le q \le Q)$ represents the set of nodes in the $\sum_1$ layer connected with the q-th product node, while the set of all product nodes connected with the n-th node in the $\sum_1$ layer is represented by $\mathcal{F}_n(1 \le n \le N)$. We can suppose an arbitrary muster italicize, and $\varphi(a)$ is the number of elements in muster italicize, and we have

$$\sum_{q=1}^{Q} \varphi(\mathcal{H}_q) = \sum_{n=1}^{N} \varphi(\mathcal{F}_n),$$

which will be used later.

We can compute the output vector $\delta \in \mathbb{R}^Q$ in the $\prod$ layer by

$$\delta_q = \prod_{i \in \mathcal{H}_q} \xi_i, 1 \le q \le Q.$$

To make convention, we denote $\prod_{i \in \mathcal{H}q} \xi_i = 1$ when $\mathcal{H}_q = \emptyset$. In the SPSNNs, the final output can be written as

$$y = g(W_0 \cdot \delta).$$

3. Batch gradient algorithm for Sigma-Pi-Sigma neural networks with elastic net regularization

3.1. Batch gradient algorithm for Sigma-Pi-Sigma neural networks

We introduce the batch gradient algorithm for SPSNNs. Let $\{X^l, O^l\}_{l=1}^L \subset \mathbb{R}^P \times \mathbb{R}$ be the given set of the training samples, where X^l denotes the *lth* input sample and the O^l is the *lth* corresponding ideal output. Let $y^l \in R$ be the real output for each input X^l. The conventional square error function can be given as

$$\begin{aligned}\widehat{E}(W) &= \frac{1}{2}\sum_{l=1}^{L}(g(W_0\cdot\delta^l)-O^l)^2\\ &= \sum_{l=1}^{L}\frac{1}{2}(g(W_0\cdot\delta^l)-O^l)^2\\ &= \sum_{l=1}^{L}g_l(W_0\cdot\delta^l),\end{aligned}$$

where $g_l(z)=\frac{1}{2}(g(z)-O^l)^2, z\in\mathbb{R}, 1\le l\le L$.

For convenience, we need to provide the following forms:

$$\begin{aligned}\delta^l &= (\delta_1^l,\delta_2^l,\cdots\delta_Q^l)\\ &= (\prod_{i\in\mathcal{H}_1}\xi_i,\prod_{i\in\mathcal{H}_2}\xi_i,\cdots\prod_{i\in\mathcal{H}_Q}\xi_i)\\ &= (\prod_{i\in\mathcal{H}_1}g(W_i\cdot X^l),\prod_{i\in\mathcal{H}_2}g(W_i\cdot X^l),\cdots,\prod_{i\in\mathcal{H}_Q}g(W_i\cdot X^l)),\end{aligned}\tag{3.1}$$

and thus, by virtue of

$$\widehat{E}(W)=\sum_{l=1}^{L}g_l(\sum_{q=1}^{Q}w_{0q}\cdot\prod_{i\in\mathcal{H}_Q}g(W_i\cdot X^l))\tag{3.2}$$

and some calculation, we can gain

$$\widehat{E}_{W_0}(W)=\sum_{l=1}^{L}g_l'(W_0\cdot\delta^l)\cdot\delta^l.$$

It follows from $\delta_q^l=\prod_{i\in\mathcal{H}_Q}g(W_i\cdot X^l)$ that

$$\frac{\partial\delta_q^l}{\partial W_n}=\begin{cases}(\prod_{i\in\mathcal{H}_q\setminus n}\xi_i)\cdot g'(W_n\cdot X^l)X^l, & if\ q\neq 1\ and\ n\in\mathcal{H}_q,\\ 0, & if\ q\equiv 1\ or\ n\notin\mathcal{H}_q.\end{cases}$$

Then, from the above equality and (3.2), we get

$$\widehat{E}_{W_n}(W)=\sum_{l=1}^{L}[g_l'(W_0\cdot\delta^l)\sum_{q=1}^{Q}(w_{0q}\cdot\frac{\partial\delta_q^l}{\partial W_n})].$$

3.2. Batch gradient algorithm for Sigma-Pi-Sigma neural networks with elastic net regularization

To simplify the network structure, we add the elastic net regularization to optimize the network on the group level. So, we can get the corresponding form of the error function

$$\begin{aligned} E(W) = \sum_{l=1}^{L} g_l(W_0 \cdot \delta^l) + \tau(\alpha(\|W_0\|_1 + \sum_{n=1}^{N} \|W_n\|_1) \\ + (1-\alpha)(\|W_0\|_2^2 + \sum_{n=1}^{N} \|W_n\|_2^2)), \end{aligned} \tag{3.3}$$

where τ and α are the tuning parameters that control the performance of the penalty term. With the gradual increase of α, the L_1 regular term dominates, and the error function gets closer to Lasso regression; with the gradual decrease of α, the L_2 regular term dominates, and the error function gets closer to ridge regression. In particular, the error function is equivalent to that of ridge regression at $\alpha = 0$, and the cost function is equivalent to that of Lasso regression at $\alpha = 1$. Let $\tau_1 = \tau \cdot \alpha$ and $\tau_2 = 2\tau \cdot (1-\alpha)$. Then, we have

$$\begin{aligned} E(W) = \sum_{l=1}^{L} g_l(W_0 \cdot \delta^l) + \tau_1(\|W_0\|_1 + \sum_{n=1}^{N} \|W_n\|_1) \\ + \frac{\tau_2}{2}(\|W_0\|_2^2 + \sum_{n=1}^{N} \|W_n\|_2^2). \end{aligned} \tag{3.4}$$

The gradient of the error function with respect to W_0 and W_n are, respectively, given as

$$E_{W_0}(W) = \sum_{l=1}^{L} g_l'(W_0 \cdot \delta^l) \cdot \delta^l + \tau_1 \frac{W_0}{\|W_0\|} + \tau_2 W_0 \tag{3.5}$$

and

$$\begin{aligned} E_{W_n}(W) = \sum_{l=1}^{L} [g_l'(W_0 \cdot \delta^l) \sum_{q \in \mathcal{F}_n} [w_{0q}(\prod_{i \in \mathcal{H}_q \backslash n} \xi_i^l) \\ \times g'(W_n \cdot X^l) \cdot X^l]] + \tau_1 \frac{W_n}{\|W_n\|} + \tau_2 W_n. \end{aligned} \tag{3.6}$$

We notice that elastic net regularization in (3.4) is combined with L_1 norm regularization and L_2 norm regularization. It is clear that (3.4) is not differentiable at the origin, which will yield the oscillation phenomenon, and we propose a smoothing approximation method to overcome this problem caused by the non-smoothness. For any limited dimensional vector $\mathbf{u}$ and a fixed constant $\gamma > 0$, we can define a smoothing function of $\|\mathbf{u}\|$ as follows:

$$h(\mathbf{u}, \gamma) = \begin{cases} \|\mathbf{u}\|, & if\ \|\mathbf{u}\| > \gamma, \\ \frac{\|u\|^2}{2\gamma} + \frac{\gamma}{2}, & if\ \|\mathbf{u}\| \le \gamma. \end{cases} \tag{3.7}$$

We use (3.7) to approximate the elastic net regularization in (3.4). Furthermore, the gradient of $h(\mathbf{u}, \gamma)$ with respect to the vector $\mathbf{u}$ is given as follows:

$$\nabla_{\mathbf{u}} h(\mathbf{u}, \gamma) = \begin{cases} \frac{\mathbf{u}}{\|\mathbf{u}\|}, & if \ \|\mathbf{u}\| > \gamma, \\ \frac{\mathbf{u}}{\gamma}, & if \ \|\mathbf{u}\| \leq \gamma. \end{cases}$$

Accordingly, the error function (3.4) can be rewritten as

$$\begin{aligned} E(W) = &\sum_{l=1}^{L} g_l(W_0 \cdot \delta^l) + \tau_1[h(W_0, \gamma) + \sum_{n=1}^{N} h(W_n, \gamma)] \\ &+ \frac{\tau_2}{2}(\|W_0\|_2^2 + \sum_{n=1}^{N} \|W_n\|_2^2). \end{aligned} \tag{3.8}$$

According to (3.8), we can get a smoothing elastic net Sigma-Pi-Sigma neural network as

$$E_{W_0}(W) = \sum_{l=1}^{L} g_l'(W_0 \cdot \delta^l) \cdot \delta_q^l + \tau_1 \nabla_{W_0} h(W_0, \gamma) + \tau_2 W_0, \tag{3.9}$$

$$\begin{aligned} E_{W_n}(W) = &\sum_{l=1}^{L} g_l'(W_0 \cdot \delta^l) \sum_{q \in \mathcal{F}_n} w_{0q} (\prod_{i \in \mathcal{H}_q \backslash n} \xi_i^l) \\ &\times g'(W_n \cdot X^l) \cdot X^l + \tau_1 \nabla_{W_n} h(W_n, \gamma) + \tau_2 W_n, \end{aligned} \tag{3.10}$$

where $l = 1, 2, \cdots, L$.

Beginning with an arbitrary initial weight vector W^0, by the following iterative formula we define the weight sequence

$$W^{k+1} = W^k + \triangle W^k, \tag{3.11}$$

$$\triangle W_0^k = -\eta E_{W_0}(W^k), \tag{3.12}$$

$$\triangle W_n^k = -\eta E_{W_n}(W^k), \tag{3.13}$$

where η represents the learning rate.

4. Simulation results

In this section, the performance of the models with no regularizer, the L_2 regularizer, 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) algorithms are compared with the smoothing elastic net regularizer algorithm (SGL) by using four examples: classification problem, parity problem, function approximation problem, and prediction problem.

4.1. Classification problem

In this example, we choose 8 benchmark data sets from the UCI machine learning repository to test the performance of the new algorithm (SGL), and compare it with the no regularizer, the L_2 regularizer, the $OL_{1/2}$, the $SL_{1/2}$, and the OGL algorithms.

Table 1 presents the main characteristics of the relevant data sets, which includes the size of datasets, attributes, categories, and sizes of the training and testing sets, where the dataset is randomly partitioned into two subsets: 70% for training and 30% for testing.

Table 1. Detailed description of the classification data sets.

Dataset	Dataset Size	Training set	Testing set	Attributes	Classes
Ecoli	336	224	112	8	7
Olitos	120	80	40	25	4
Seeds	210	147	63	7	3
Iris	150	105	45	4	3
Wine	178	120	58	13	3
Liver	345	240	105	7	2
Sonar	208	138	70	60	2
Diabetes	768	526	242	8	2

As described at the beginning of this paper, we learn the structure of SPSNNs (see Figure 1). Then, we select $P = 13$, $N = 4$, $Q = 16$, and 1, representing the number of the nodes of the input, the $\sum_1$, the $\prod$, and the $\sum_2$ layers, respectively. For each learning algorithm, the initial weights are randomly selected in the interval $[-0.5, 0.5]$, the learning rate η is 0.0028, and the regular factor τ is 0.001, and we conduct 20 trials for every data set to compare the performance of different algorithms.

To assess the performance of the smoothing elastic net regularizer, based on each data set, we compare the number of remaining hidden neurons after pruning (RNN), the training accuracy testing accuracy, and training time for each algorithm, and all experimental results are recorded in Table 2. From the table, it can be observed that the training accuracy has slightly improved, while the testing accuracy has increased by approximately 1% to 3%. We can find our proposed smoothing elastic net regularizer is superior to the no regularizer, the L_2 regularizer, the original $L_{1/2}$ regularizer, the smoothing $L_{1/2}$ regularizer, and the original group Lasso regularizer algorithms.

In addition, we have also compared our approach with other existing methods. In [45], the authors considered the group Lasso regularization method on the Sigma-Pi-Sigma neural network. In [46], the authors applied the group $L_{1/2}$ regularization term on high-order neural networks. Our proposed elastic net regularization method is on par with these approaches.

Table 2. Performance comparison for classification problems.

Dataset	Algorithm	*RNN*	Training accuracy	Testing accuracy	Training time(s)
Ecoli	*NoPenalty*	13.30	0.9761	0.9082	17.2756
	L_2	13.00	0.9747	0.8980	**16.7845**
	$OL_{1/2}$	12.50	0.9749	0.9191	18.1372
	$SL_{1/2}$	11.80	0.9771	0.9304	16.8050
	OGL	12.20	0.9749	0.9191	18.3792
	SGL	**11.50**	**0.9780**	**0.9314**	17.3763
Olitos	*NoPenalty*	13.00	0.9573	0.8914	29.9742
	L_2	12.33	0.9592	0.9053	**29.3393**
	$OL_{1/2}$	12.00	0.9604	0.9160	30.6111
	$SL_{1/2}$	11.67	0.9589	0.9245	30.3067
	OGL	12.00	0.9617	0.9264	33.0595
	SGL	**11.00**	**0.9628**	**0.9355**	30.5098
Seeds	*NoPenalty*	13.33	0.9737	0.9522	13.4081
	L_2	12.67	0.9761	0.9554	13.5387
	$OL_{1/2}$	12.33	0.9791	0.9582	13.8205
	$SL_{1/2}$	11.67	0.9797	0.9676	13.2730
	OGL	12.33	0.9792	0.9629	14.7718
	SGL	**11.33**	**0.9813**	**0.9749**	**12.7701**
Iris	*NoPenalty*	13.67	0.9715	0.9296	13.4447
	L_2	13.00	0.9719	0.9390	**13.4420**
	$OL_{1/2}$	12.67	0.9723	0.9458	14.3637
	$SL_{1/2}$	12.33	0.9743	0.9554	13.5318
	OGL	12.33	0.9748	0.9522	16.1023
	SGL	**11.00**	**0.9791**	**0.9629**	14.2630
Wine	*NoPenalty*	12.67	0.9872	0.9729	20.4322
	L_2	12.67	0.9892	0.9753	20.4173
	$OL_{1/2}$	12.00	0.9896	0.9770	21.3012
	$SL_{1/2}$	11.50	0.9911	0.9814	20.5545
	OGL	11.67	0.9906	0.9798	21.1104
	SGL	**10.50**	**0.9915**	**0.9833**	**20.1607**
Liver	*NoPenalty*	13.33	0.9937	0.9823	15.5081
	L_2	12.67	0.9943	0.9838	**15.4588**
	$OL_{1/2}$	12.33	0.9947	0.9858	16.4440
	$SL_{1/2}$	11.33	0.9951	0.9861	15.9798
	OGL	11.00	0.9948	0.9868	17.4374
	SGL	**10.33**	**0.9962**	**0.9902**	16.1645
Sonar	*NoPenalty*	12.67	0.9825	0.9756	12.6535
	L_2	12.67	0.9831	0.9787	12.1906
	$OL_{1/2}$	12.33	0.9860	0.9830	12.9125
	$SL_{1/2}$	12.00	0.9909	0.9852	12.5690
	OGL	12.33	0.9892	0.9849	13.7648
	SGL	**11.67**	**0.9918**	**0.9860**	**11.7338**
Diabetes	*NoPenalty*	13.00	0.9925	0.9937	17.7933
	L_2	12.50	0.9936	0.9947	**17.1156**
	$OL_{1/2}$	11.67	0.9954	0.9950	17.4822
	$SL_{1/2}$	11.33	0.9967	0.9955	17.1170
	OGL	11.67	0.9961	0.9953	18.4761
	SGL	**11.33**	**0.9978**	**0.9961**	17.3682

4.2. 5-bit parity problem

For the parity problem, there is an input set of 2^n samples in n-dimensional space, and every sample is an n-bit binary vector. We consider a 5-bit parity problem which has an input set with 2^5 samples in 5-dimensional space; the ideal output equals to 1 if the number of 1 in the samples is odd, otherwise

it equals to zero. Here, using the above method, we test the performance of our proposed smoothing elastic net regularizer.

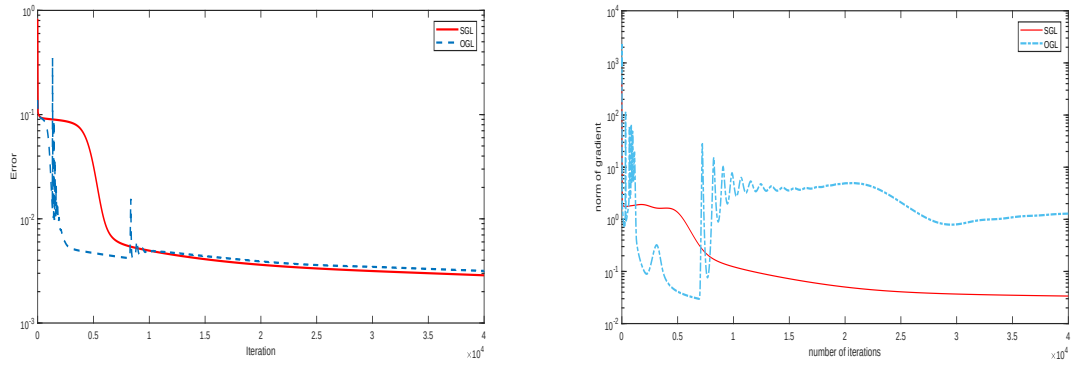
Similarly, we can study the structure of SPSNNs. We select $P = 13$, $N = 4$, $Q = 16$, and 1 for the number of the nodes of the input, $\sum_1$, $\prod$, and $\sum_2$ layers, separately. In the interval $[-0.5, 0.5]$, the initial weights are randomly selected, the learning rate η is 0.0045, and the regular factor τ is 0.001. For each learning algorithm we carry out 20 experiments, and we train up to 40,000 steps or we stop once the error is less than $1e-4$. So, as to assess the sparsity and convergence of the smoothing elastic net regularizer, we compare the average error (AVE) and the number of remaining hidden neurons after pruning (RNN) with the no regularizer, the L_2 regularizer, the original $L_{1/2}$ regularizer, the smoothing $L_{1/2}$ regularizer, the original group Lasso regularizer, and the smoothing elastic net regularizer, which are listed in Table 3.

Table 3. Emulation results for 5-bit parity problem.

Learning Methods	AVE	RNN
$NoPenalty$	0.004433	17.00
L_2	0.003967	17.00
$OL_{1/2}$	0.003929	17.14
$SL_{1/2}$	0.004033	17.33
OGL	0.003925	17.00
SGL	0.003471	16.71

The results show that the proposed smooth elastic net regularizer outperforms the no regularizer, the L_2 regularizer, the original $L_{1/2}$ regularizer, the smoothing $L_{1/2}$ regularizer, and the original group Lasso regularizer.

Figure 2(a) shows the error performance of the original group Lasso regularizer and smoothing elastic net regularizer via the 5-bit parity problem. Figure 2(b) shows that the norm of the gradient curve of the error function, based on the 5-bit parity problem, approaches a small positive constant. This indicates that the smoothing elastic net regularizer removes the oscillation of occurring in the original group Lasso regularizer in the learning process.



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

Figure 2. The performance results based on 5-bit parity problem with OGL and SGL.

4.3. Function approximation problem

In this example, we study the multi-dimensional Gabor function to compare the approximation performance of the above algorithms.

$$k(x,y) = \frac{1}{2\pi(0.5)^2} exp(-\frac{x^2+y^2}{2(0.5)^2}) cos(2\pi(x+y)).$$

As described at the beginning of this paper, we learn the structure of the SPSNNs. Then, we select $P = 13$, $N = 4$, $Q = 16$, and 1 for the number of the nodes of the input layer, $\sum_1$, $\prod$, and $\sum_2$ layers, separately.

In this experiment, we select 169 training samples from an evenly spaced 6×6 grid on the square $-0.5 \le x \le 0.5$ and $-0.5 \le y \le 0.5$. In the interval $[-0.5, 0.5]$, the initial weights are randomly selected, the learning rate η is 0.0028, and the regular factor τ is 0.001. For each learning algorithm we carry out 20 experiments, and we train up to 40,000 times or until the error is less than $1e-4$ stop iterations.

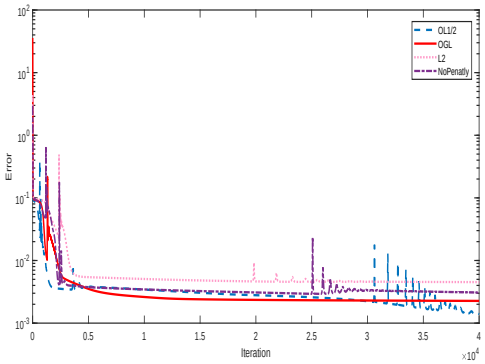
To assess the sparsity and convergence of the smoothing elastic net regularizer, we compare the average error (AVE) and the number of remaining hidden neurons after pruning (RNN) with the no regularizer, the L_2 regularizer, the original $L_{1/2}$ regularizer, the smoothing $L_{1/2}$ regularizer, the original group Lasso regularizer, and the smoothing elastic net regularizer, which are shown in Table 4. We can see our proposed smoothing elastic net regularizer is superior to the no regularizer, the L_2 regularizer, the original $L_{1/2}$ regularizer, the smoothing $L_{1/2}$ regularizer, and the original group Lasso regularizer.

Table 4. Emulation results for identifying the Gabor function.

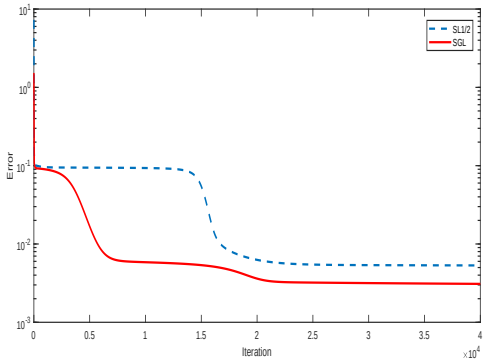
Learning Methods	AVE	RNN
$NoPenalty$	0.003940	18.00
L_2	0.003560	18.00
$OL_{1/2}$	0.003783	17.67
$SL_{1/2}$	0.004486	17.37
OGL	0.003523	16.87
SGL	0.003286	16.14

For each learning algorithm, we show the error function and the norm of gradient of one of the 20 experiments after 40,000 epochs in Figures 3–5. Figure 3(a) shows the oscillation phenomenon of no regularizer, the L_2 regularizer,the original $L_{1/2}$ regularizer and the original group lasso regularizer. Figure 3(b) shows the error curve of the smoothing $L_{1/2}$ regularizer and the smoothing elastic net regularizer. Figure 4(a) shows the norm of gradient curve of the no regularizer, the L_2 regularizer, the original $L_{1/2}$ regularizer, and the original group Lasso regularizer. Figure 4(b) shows the norm of the gradient curve of the smoothing $L_{1/2}$ regularizer, and the smoothing elastic net regularizer. Obviously, it approaches to a small positive constant. Figure 5 shows a typical performance for one of 20 experiments, and we can see that it has a good approximation effect compared with other algorithms. In each learning algorithm for the same parameters, we get the corresponding results. We can see the learning method with the smoothing elastic net regularizer converges faster than

other learning methods, and the smoothing elastic net regularizer method overcomes the numerical oscillation phenomenon. It also shows that during the iterative process the error function curves are monotonically decreasing and converge to zero, which also validates our theoretical results.

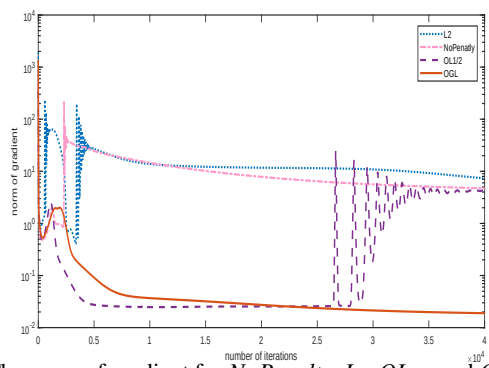


(a) The curve of error function for *NoPenalty*, L_2, $OL_{1/2}$ and *OGL*

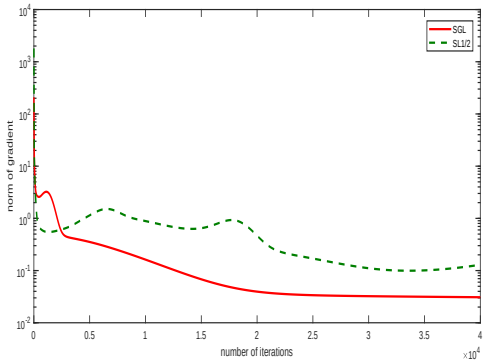


(b) The curve of error function for $SL_{1/2}$ and *SGL*.

Figure 3. The performance results of the error function for the above algorithms.



(a) The norm of gradient for *NoPenalty*, L_2, $OL_{1/2}$ and *OGL*.



(b) The norm of gradient for $SL_{1/2}$ and *SGL*

Figure 4. The performance results of the gradient norm for the above algorithms.

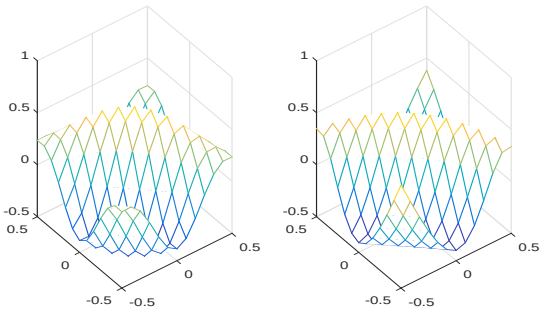


Figure 5. The approximation result of the smoothing elastic net regularizer.

4.4. Prediction problem

To verify the effectiveness of the algorithm further, this part takes an interval shield tunneling project of Metro Line 9 in Zhengzhou City, China, as an example. In order to monitor the impact of the subway shield tunneling process on the surface buildings and structures, 10-meter intervals are used to set up settlement observation points in advance in each shield tunneling section, and the surrounding structures and buildings are monitored. JGC1, the closest settlement observation point of the signal tower to the right line of the shield structure, is selected as the objective of study, and the Leica DNA03 level is used to collect data 10 days before the shield structure is excavated to the closest point of JGC1, for a total of 40 days, and the frequency of observation is once a day. In this experiment, we use the data of the first 30 days as the training data set and the data of the last 10 days as the test data set (see Table 5).

Table 5. Measured settlement of metro shield at point JGC1.

Time	JGC1 (mm)	Time	JGC1 (mm)
2015.3.18	+0.21000	2015.3.31	0.003286
2015.3.19	-0.13000	2015.4.2	+0.10000
2015.3.20	+0.11000	2015.4.3	-0.02000
2015.3.21	-0.06000	2015.4.4	+0.08000
2015.3.22	-0.23000	2015.4.5	-0.16000
2015.3.23	-0.23000	2015.4.6	+0.06000
2015.3.24	-0.15000	2015.4.7	-0.18000
2015.3.25	-0.03000	2015.4.8	+0.28000
2015.3.26	0.003286	2015.4.9	-0.07000
2015.3.27	0.003286	2015.4.10	+0.07000
Time	JGC1 (mm)	Time	JGC1 (mm)
2015.4.11	+0.01000	2015.4.22	+0.01000
2015.4.12	-0.01000	2015.4.23	0.00000
2015.4.13	-0.15000	2015.4.24	+0.19000
2015.4.14	+0.08000	2015.4.25	-0.21000
2015.4.15	-0.09000	2015.4.26	+0.10000
2015.4.16	+0.14000	2015.4.27	-0.12000
2015.4.17	-0.02000	2015.4.28	+0.05000
2015.4.18	-0.26000	2015.4.29	+0.15000
2015.4.19	+0.42000	2015.4.30	-0.17000
2015.4.20	-0.22000	2015.5.1	-0.07000

In this experiment, we learn the structure of the SPSNNs. Then, we select $P = 5$, $N = 4$, $Q = 16$, and 1 for the number of the nodes of the input layer, $\sum_1$, $\sum_2$, and the output layer, respectively. We use the sigmod activation function at the $\sum_1$ and output layers, respectively, and our stopping criteria in this experiment is an error of less than 1×10^{-5} or 5000 iterations.

Figure 6 is the error curve of the training set with 5000 iterations, in which red is the error curve without the regularization term and blue is the error curve with the smoothing elastic net regularization,

it can be obtained that the error of the network with the smoothing elastic net regularization decreases faster, and after 500 iterations, the error is smaller than that without the regularization term, which precisely verifies the theoretical results of this paper and the effectiveness of the proposed algorithm.

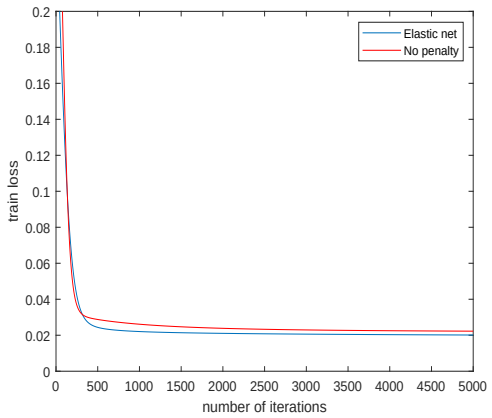


Figure 6. The curve of the error function for no penalty and for the smoothing elastic net.

5. Conclusions

In this paper, a new batch gradient algorithm for SPSNNs with an L_1 plus L_2 regularization algorithm is proposed as an effective weight pruning technique. It can handle multi-output regression and multi-class classification problems within a unified framework. This algorithm obtains good performance in both Lasso and ridge regression, penalizing the weights by reducing the weight vectors to zero, which is more efficient than other various pruning strategies. Moreover, the theoretical results and the advantages of this algorithm are also illustrated by numerical experiments.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Author contribution

Conceptualization, methodology, original draft preparation, J. Jiao; software, editing, K. Su.

Conflict of interest

All authors declare there is no conflict of interest.

References

1. C. K. Li, A sigma-pi-sigma neural network(SPSNN), *Neural Processing Letters,* **17** (2003), 1–19. https://doi.org/10.1023/A:1022967523886

2. Q. W. Fan, F. J. Zheng, X. D. Huang, D. P. Xu, Convergence Analysis for Sparse Pi-Sigma Neural Network Model with Entropy Error Function, *International Journal of Machine Learning and Cybernetics,* (2023), 1–12. https://doi.org/10.1007/s13042-023-01901-x
3. Q. W. Fan, L. Liu, Q. Kang, L. Zhou, Convergence of Batch Gradient Method for Training of Pi-Sigma Neural Network with Regularizer and Adaptive Momentum Term, *Neural Process. Lett.,* **4** (2023), 55. https://doi.org/10.1007/s11063-022-11069-0
4. J. C. Valle-Lisboa, F. Reali, H. Anastasia, E. Mizraji, Elman topology with sigma-pi units: An application to the modeling of verbal hallucinations in schizophrenia, *Neural Netw.,* **18** (2005), 863–877. https://doi.org/10.1016/j.neunet.2005.03.009
5. C. Weber, S. Wermter, A self-organizing map of sigma-pi units, *Neurocomputing,* **70** (2007), 2552–2560. https://doi.org/10.1016/j.neucom.2006.05.014
6. Z. M. Chen, K. Niu, L. Li, Research on adaptive trajectory tracking algorithm for a quadrotor based on backstepping and the Sigma-Pi neural network, *Int. J. Aerosp. Eng.,* **2019** (2019), 1–9. https://doi.org/10.1155/2019/1510341
7. M. Fallahnezhad, M. H. Moradi, S. Zaferanlouei, A hybrid higher order neural classier for handling classfication problems, *Expert Syst. Appl.,* **38** (2011), 386–393. https://doi.org/10.1016/j.eswa.2010.06.077
8. Y. B. Wang, T. X. Li, J. Y. Li, W. C. Li, Analysis on the performances of sparselized sigma-pi networks, in: Proceedings of the World Multi-conference on Systemics, *Cybernetics and Informatics, Florida, USA,* **5** (2004), 394–398.
9. B. Dario, M. D. Fernando, A survey of artificial neural network training tools, *Neural Comput. Appl.,* **23** (2013), 609–615. https://doi.org/10.1007/978-3-540-77465-5-13
10. L. Xu, J. S. Chen, D. F. Huang, Analysis of boundedness and convergence of online gradient method for two-Layer feedforward neural networks, *IEEE Trans. Neural Netw. Learn. Syst.,* **24** (2013), 1327–1338. https://doi.org/10.1109/TNNLS.2013.2257845
11. Q. W. Fan, Z. W. Zhang, X. D. Huang, Parameter conjugate gradient with secant equation based Elman neural network and its convergence analysis, *Adv. Theor. Simul.,* 2022, 1–12. https://doi.org/10.1002/adts.202200047
12. J. Larsen, C. Svarer, L. N. Andersen, Adaptive regularization in neural network modeling, *LNCS,* **7700** (2012), 111–130. https://doi.org/10.1007/3-540-49430-8-6
13. H. T. Huynh, Y. Won, Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks, *Pattern Recognit. Lett.,* **32** (2011), 1930–1935. https://doi.org/10.1016/j.neucom.2016.04.043
14. S. E. Fahlman, C. Lebiere, The cascade-correlation learning architecture, 1990.
15. E. D. Karnin, A simple procedure for pruning back-propagation trained neural networks, *IEEE Trans. Neural Netw.,* **1** (1990), 239–242. https://doi.org/10.1109/72.80236
16. R. Reed, Pruning algorithms-a survey, *IEEE Trans. Neural Netw.,* **4** (1993), 740–747. https://doi.org/10.1109/72.248452
17. H. G. Han, J. F. Qiao, A structure optimisation algorithm for feedforward neural network construction, *Neurocomputing,* **99** (2013), 347–357. https://doi.org/10.1016/j.neucom.2012.07.023

18. A. B. Nielsen, L. K. Hansen, Structure learning by pruning in independent component analysis, *Neurocomputing,* **71** (2008), 2281–2290. https://doi.org/10.1016/j.neuron.2014.05.035
19. J. F. Qiao, Y. Zhang, H. G. Han, Fast unit pruning algorithm for feed-forward neural network design, *Appl. Math. Comput.,* **205** (2008), 662–667. https://doi.org/10.1016/j.amc.2008.05.049
20. J. L. Li, F. Jiao, J. C. Fang, J. C. Cheng, Temperature error modeling of RLG based on neural network optimized by PSO and regularization, *IEEE Sens. J.,* **14** (2014), 912–919. https://doi.org/10.1109/JSEN.2013.2290699
21. J. P. Donate, X. D. Li, G. G. Sa'nchez, A. S. Miguel, Time series forecasting by evolving artificial neural networks with genetic algorithms,differential evolution and estimation of distribution algorithm, *Neural Comput. Appl.,* **22** (2013), 11–20. https://doi.org/10.1007/s00521-011-0741-0
22. O. Ludwig, Eigenvalue decay: A new method for neural network regularization, *Neurocomputing,* **124** (2014), 33–42. https://doi.org/10.1016/j.neucom.2013.08.005
23. S. U. Ahmed, M. Shah, K. Murase, A lempel-ziv complexity-based neural network pruning algorithm, *Int. J. Neural Syst.,* **21** (2011), 427–441. https://doi.org/10.1142/S0129065711002936
24. T. T. Pan, J. H. Zhao, W. Wu, J. Yang, Learning imbalanced datasets based on SMOTE and Gaussian distribution, *Inf. Sci.,* **512** (2020), 1214–1233. https://doi.org/10.1016/j.ins.2019.10.048
25. I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Cambridge, MA, USA: MIT Press, 2016.
26. G. E. Hinton, Deterministic Boltzmann learning performs steepest descent in weight-space, *Neural Comput.,* **1** (1989), 143–150. https://doi.org/10.7551/mitpress/3349.003.0007
27. J. Sum, C. S. Leung, K. Ho, Convergence analyses on on-line weight noise injection-based training algorithms for MLPs, *IEEE Trans. Neural Netw. Learn. Syst.,* **23** (2012), 1827–1840. https://doi.org/10.1109/TNNLS.2012.2210243
28. P. May, E. Zhou, A comprehensive evaluation of weight growth and weight elimination methods using the tangent plane algorithm, *Int. J. Adv. Comput. Sci. Appl.,* **4** (2013), 149–156. https://doi.org/10.14569/IJACSA.2013.040621
29. J. E. Moody, T. S. Rognvaldsson, Smoothing regularizers for projective basis function networks, *Proc. Adv. Neural Inf. Process. Syst.,* **9** (1997), 585–591.
30. Z. Chen, S. Haykin, On different facets of regularization theory, Neural Comput., **14**(12), 2791–2846. https://doi.org/10.1162/089976602760805296
31. Q. W. Fan, Q. Kang, J. M. Zurada, T. W. Huang, D. P. Xu. Convergence analysis of online gradient method for High-Order neural networks and their sparse optimization, *IEEE T. Neur. Net. Lear.,* 2023. https://doi.org/10.1109/TNNLS.2023.3319989
32. L. Zhou, Q. W. Fan, X. D. Huang, Y. Liu, Weak and strong convergence analysis of elman neural networks via weight decay regularization, *Optimization,* **72** (2023), 2287–2309. https://doi.org/10.1080/02331934.2022.2057852
33. M. G. Augasta, T. Kathirvalavakumar, Pruning algorithms of neural networks-a comparative study, *Central Eur. J. Comput. Sci.,* **3**(2013), 105–115. https://doi.org/10.2478/s13537-013-0109-x
34. W. Wu, H. M. Shao, Z. X. Li, Convergence of batch BP algorithm with penalty for FNN training, *Neural Inf. Process.,* **4232** (2006), 562–569. https://doi.org/10.1007/11893028-63

35. J. Wang, W. Wu, J. M. Zurada, Computational properties and convergence analysis of BPNN for cyclic and almost cyclic learning with penalty, *Neural Netw.,* **33** (2012), 127–135. https://doi.org/10.1016/j.neunet.2012.04.013
36. K. Saito, S. Nakano, Second-order learning algorithm with squared penalty term, *Neural Comput.,* **12** (2000), 709–729.
37. H. Zhang, W. Wu, M. Yao, Boundedness and convergence of batch backpropagation algorithm with penalty for feedforward neural networks, *Neurocomputing,* **89** (2012), 141–146. https://doi.org/10.1016/j.neucom.2012.02.029
38. X. Y. Chang, Z. B. Xu, H. Zhang, J. J. Wang, Y. Liang, Robust regularization theory based on L_q $(0 < q < 1)$ regularization: the asymptotic distribution and variable selection consistence of solutions, *Sci. China,* **40** (2010), 985–998.
39. B. K. Natarajan, Sparse approximate solutions to linear systems, *SIAM J. Comput.,* **24** (1995), 227–234. https://doi.org/10.1137/S0097539792240406
40. R. Tibshirani, Regression shrinkage and selection via the Lasso, *J. R. Stat. Soc. Ser. B.,* **58** (1996), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x
41. H. Bilal, A. Kumar, B. Yin, Pruning filters with L_1-norm and capped L_1-norm for CNN compression, *Appl. Intell.,* **51** (2021), 1152–1160. https://doi.org/10.1007/s10489-020-01894-y
42. H. J. Rong, Y. S. Ong, A. H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, *Neurocomputing,* **72** (2008), 359–366. https://doi.org/10.1016/j.neucom.2008.01.005
43. J. M. Martinez-Martinez, P. Escandell-Montero, E. Soria-Olivas, J. D. Martin-Guerrero, R. Magdalena-Benedito, J. Gmez-Sanchis, Regularized extreme learning machine for regression problems, *Neurocomputing,* **74** (2011), 3716–3721. https://doi.org/10.1016/j.neucom.2011.06.013
44. C. De Mol, E. De Vito, L. Rosasco, Elastic-net regularization in learning theory, *J. Complex.,* **25** (2009), 201–230. https://doi.org/10.1016/j.jco.2009.01.002
45. Q. Kang, Q. W. Fan, J. M. Zurada, Deterministic convergence analysis via smoothing group Lasso regularization and adaptive momentum for sigma-pi-sigma neural network, *Inform. Sciences,* **553** (2021), 66–82. https://doi.org/10.1016/j.ins.2020.12.014
46. Q. Kang, Q. W. Fan, J. M. Zurada, T. W. Huang, A pruning algorithm with relaxed conditions for high-order neural networks based on smoothing group $L_{1/2}$ regularization and adaptive momentum, *Knowledge-Based Syst.,* **257** (2022), 109858. https://doi.org/10.1016/j.knosys.2022.109858

AIMS Press