Sensors & Transducers, Vol. 260, Issue 1, May 2023, pp. 14-23

www.sensorsportal.com

Sensors & Transducers

Published by IFSA Publishing, S. L., 2023
http://www.sensorsportal.com

Pruning Feedforward Polynomial Neural with Smoothing Elastic Net Regularization

[1,2 *] Khidir Shaib Mohamed

[1]Department of Mathematics, College of Sciences and Arts in Uglat Asugour,
Qassim University, Buraydah 51452, Kingdom of Saudi Arabia
[2]Department of Mathematics and Computer, College of Science, Dalanj University,
Dalang P.O. Box 14, South Kordofan, Sudan
E-mail: K.Idris@qu.edu.sa or khshm7@yahoo.com

Received: 21 January 2023 Accepted: 16 March 2023 Published: 23 May 2023

Abstract: Gradient methods are preferred for training and pruning neural networks because regularization terms are primarily intended to remove redundant weights from neural networks. Many machine learning libraries use elastic net regularization (ENR) also called double regularization, which is a combination of L_1 and L_2 regularizations which tends to have a grouping effect in which correlated input features are given equal weights. This paper proposes a batch gradient method with smoothing elastic net regularization for pruning feedforward polynomial neural networks (FFPNNs), especially pi-sigma neural networks (PSNNs). Unfortunately, since elastic net regularization contains the 1-norm, is non-differentiable, and does not produce an NP-hard problem, it is not possible to use the gradient method directly. We attempt to replace the 1-norm and end up with the smoothing elastic net regularization in order to overcome this obstacle by using a differentiable and continuous function. The monotonicity theorem and two convergence theorems, including a weak convergence and a strong convergence, are established under this circumstance. The validity of the proposed theorems is supported by the experimental findings. According to the numerical results, the smoothing double regularization improved generalization performance and accelerated the learning process.

Keywords: Convergence, Batch gradient method, Smoothing elastic net regularization, Pi-sigma neural networks

1. Introduction

Higher-order polynomial neural networks (HOPNNs) have been developed with intention to enhance the nonlinear descriptive capacity of the feedforward multilayer perceptron networks. HOPNNs are networks that utilize higher combinations of inputs. HOPNNs have been successfully applied to a variety of real-world tasks, including prediction, classification, signal processing, image recognition, and particularly in covid-19, business, and for trading the EUR/USD exchange rate [1-3], respectively. In a research paper published by Ghosh and Shin [4], they introduced the pi-sigma neural network, a type of high-order neural network, which has shown to be effective in predicting patterns and approximating functions. Since the weights between hidden units and outputs are fixed at 1, the model has multiplication neurons in output unit layers. A neural network of this type uses the product of sums, has a regular structure, is much faster at learning, and can be incrementally enlarged to achieve a desired level of complexity. Data is essential for building learning models and can be used in the modeling pipeline in two ways. In batch learning, the data is at rest, while in online learning, the data flows into the

learning algorithm in streams [5], and this paper flow the first approach. The pi-sigma neural network (PSNN) joined a gradient descent algorithm is capable to dispose of the nonlinear problems with much more powerful mapping and higher computational speed [6, 7]. The terms of PSNNs are accepted to address several difficult different problems: such as time series forecasting [8], for decade progress [9], based on sine cosine optimization algorithm [10], forecasting [11], and improved spotted hyena optimizer with space transformational search [12].

In [13], Zou and Hastie introduced a novel elastic net regularization (double regularization) method by extending the L_1 regularization by adding L_2 regularization for the variable selection method in order to overcome the limitation of the L_1 regularization. This term has been used in several studies for example as used for portfolios with risk minimization [14], high-dimensional longitudinal data [15], and discriminate analysis classifiers with automatic parameter selection [16]. The double regularization algorithm is widely used in practice and is implemented in many libraries for machine learning. The modified error function joins double regularization, which takes the following form:

$$E(W) = \check{E}(W) + \lambda_1 \|W\|_1 + \lambda_2 \|W\|_2^2, \quad (1)$$

where $\check{\mathrm{E}}(\mathrm{W})$ is the stander error function depending on the weights W, $\|\cdot\|_2$ and $\|\cdot\|_1$ denotes 2-norm and 1-norm, respectively, λ_1 and λ_2 are regularization parameters. In generally many regularization types take the form of the q-norm of the weights W of the network as

$$\|W\|_q = (\sum_{m=1}^{M} |w_m|^q)^{\frac{1}{q}} \quad (2)$$

Noting $\|W\|_q$ is the q-norm where $(0 \le q \le \infty)$, let $\mathrm{q} = 0$ is $\|W\|_0$ is known 0-norm, while $\mathrm{q} = 1/2$ is $\|W\|_{1/2}$ is referred to $1/2$ −norm. Neural network training using the gradient method has been commonly applied, but sometimes, the training process is poor and overfitting. The idea of adding a regularization term into the error has become common to improve the generalization performance of the network and to make the network weights become smaller during the training. Various types of regularization terms have been used in many applications of neural network areas. L_2 and L_1 are the most common terms used which include 2-norm and 1-norm, respectively. The L_2 regularization also called weight decay may be the most popular regularization technique used [17-19]. While the L_1 regularization has become widespread for improving the sparse solution [20, 21]. Sparse optimization involving the L_0 regularization in objective function has a wide application in many fields. AIC and BIC [22], well-known model selection criteria, are special cases of L_0 regularization. In [23], a novel $L_{1/2}$ regularization has been providing more sparsity than the L_0 and L_1 regularizations. However, since L_0, L_1, and L_1 regularization norms of weights are non-differentiable; we cannot incorporate it directly as a regularization term into the objective function. To get rid of this dilemma, many researchers have begun to use a smoothing technical that removes the inability of a function to be a string, for example (cf. [24, 25]). We try to investigate the proposal of this study which is related to these ideas.

In Section 2, we show the batch gradient method with elastic net regularization (BGENR), and the gradient descent algorithm with smoothing double regularization (BGSENR). We selected convergence results are given in Section 3. We present numerical results in Section 4. The proofs of the convergence results in Section 5. Finally, conclusion summary of this work in Section 6.

2. BGSENR

PSNN consists of an input unit, a single hidden layer of linear summation units and product units in the output layer are p, n and 1 respectively. The term pi-sigma ($\Pi\Sigma$) use products of sums of input components. PSNNs have only one unit of adjustable weights, the weights of the output unit are commonly fixed at 1 (see Fig. 1). The weight vector connecting the input unit and the k −th summing unit, introduced by $w_j = (w_{ji}, w_{j2}, \ldots, w_{jp})T \in \mathbb{R}^P$, and $\mathrm{w} = (\mathrm{w}_1^{\mathrm{T}}, \mathrm{w}_2^{\mathrm{T}}, .., \mathrm{w}_{\mathrm{n}}^{\mathrm{T}}) \in \mathbb{R}^{\mathrm{nP}}$, $(j = 1,2, \cdots, n)$. Let $g: \mathbb{R} \to \mathbb{R}$ be a given activation function. For any given input x and weight w, the output of the PSNN is

$$y = g[\prod_{j=1}^{n} (\sum_{i=1}^{p} (w_{ji} x_i))] = g[\prod_{j=1}^{n} (w_j \cdot x)], \quad (3)$$

where $w_j \cdot x$ represents the inner product of w_j and x. The topological structure of PSNN algorithm is shown below (Fig. 1).

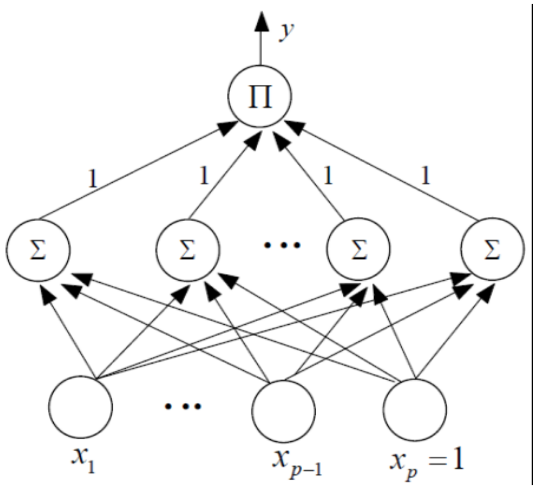

Fig. 1. Structure of pi-sigma neural network.

Let $\{x^l, O^l\}_{l=1}^L \subset \mathbb{R}^p \times \mathbb{R}$ be the set of training samples with O^l is the desired objective output for the input x^l. For any fixed weight w, the output error function with elastic net regularization (ENR) is defined as

$$E(w) = \frac{1}{2}\sum_{l=1}^{L}[O^l - g(\prod_{j=1}^{n}(w_j \cdot x^l))]^2 + \lambda_1\|\mathrm{W}\|_1 + \lambda_2\|W\|^2 \quad (4)$$

where $\|W\|^2 = \sum_{j=1}^{q} w_j^2$, $\|W\|_1 = \sum_{j=1}^{q}|w_j|$, $\|\cdot\|$ is the normal Euclidean norm, and $|\cdot|$ is the normal absolute value function and λ_1, λ_2 are regularization parameters. In above equation a special case where $\lambda_1 = \lambda$, $\lambda_2 = 0$ is known L_1 regularization method (BGL_1), while $\lambda_1 = 0$, $\lambda_2 = \lambda$ is L_2 regularization method (BGL_2).

Since the 1-norm of weights includes the absolute value which means is non-differentiable; we cannot incorporate it directly as a regularization term into the objective function. To get rid of this problem, we try to use a differential and continuous function $f(t)$ instead of an absolute value as in the above formula. Specifically, the following piecewise polynomial function is used in [24, 25] as follows:

$$f(t) = \begin{cases} |t|, & |t| \geq \gamma \\ -\frac{t^4}{8\gamma^3} + \frac{3t^2}{4\gamma} + \frac{3\gamma}{8}, & |t| < \gamma, \end{cases} \quad (5)$$

Remark 1. *The function $f(t)$ is twice continuously differentiable on $\mathbb{R}$. Moreover, $f(t), f'(t)$, and $f''(t)$ are uniformly bounded on $\mathbb{R}$, $\mathbb{N} = 3\backslash 2\mathcal{B}$, and $\gamma > 0$ is a small positive constant. This function provides the following advantages:*

1) $f(t) \to \left[\frac{3\gamma}{8}, +\infty\right)$
2) $f'(t) \to [-1, 1]$
3) $f''(t) \to \left[0, \frac{3}{2\gamma}\right]$.

Then the new error function with smoothing elastic net regularization (SENR) is as follows:

$$E(w) = \frac{1}{2}\sum_{l=1}^{L}[O^l - g(\prod_{j=1}^{n}(w_j \cdot x^l))]^2 + \lambda_1 \sum_{j=1}^{n} w_j^2 + \lambda_2 \sum_{j=1}^{n} f(w_j). \quad (6)$$

The purpose of above equation is to obtain w^* such that

$$E(w) = \min E(w^*) \quad (7)$$

The gradient descent algorithm is often used to solve this type of problem. The weights can be trained in two different ways, referred to as batch and online trainings, respectively ([26, 27]). The weights are changed in the initial approach after the network is shown each training pattern. In the second, the weights are changed after the presentation of the entire training set while the error accumulates over an epoch. This study employs the batch training methodology. Starting from an arbitrary initial value w^0, the updates the weights w^m iteratively by

$$w_j^{m+1} - w_j^m = \Delta w_j^m = -\eta E_{w_j}(w^m), \quad (8)$$

where $E_{w_j}(w^m) = \frac{\partial E(w^m)}{\partial w_j^m}$ and

$$\Delta w_j^m = -\eta[\sum_{l=1}^{L} \delta_l(\prod_{j=1}^{n}(w_j^m \cdot x^l)) \prod_{\substack{k=1 \\ k\neq j}}^{n}(w_k^m \cdot x^l)\, x^l + 2\lambda_1 w_j^m + \lambda_2 f'(w_j^m)],$$
$$j = 1,2,\ldots,J,\ m = 0,1,2,\cdots,$$

where $\delta_l(t) = -[O^l - g'(t)]g'(t)$, $\eta > 0$ is learning rate.

3. Main Results

The following propositions will later be applied to prove the convergence theorem.

Proposition 1. $|\delta_j(t)|, |\delta_j'(t)|, |g(t)|, |g'(t)|, |g''(t)| \leq C_0$ *are uniformly bounded for* $t \in \mathbb{R}$.

Proposition 2. $max\{|x^l|, |w_j^m \cdot x^l|\} \leq C_0$.

Proposition 3. *The learning rate* η *and the regularization parameters* λ_1, λ_2 *are chosen to satisfy* $0 < \eta < \frac{1}{\lambda_1 + \lambda_2\mathbb{N} + C}$, *and* $C = C_2 + C_3$.

Proposition 4. *There exists a compact set* φ *such that* $w^m \in \varphi$ *and* $\varphi_0 = \{w \in \varphi : E_w(w) = 0\}$ *is the contains finite points.*

Theorem 1. *Let the error function* $E(w)$ *be defined by Equation (6) and the weight* $\{w^m\}_{m=0}^{\infty}$ *is generated by the iteration Equation (8) for a random initial value. If propositions 1 to 3 are correct, then there are the following estimates:*

I. $E(w^{m+1}) \leq E(w^m)$, $m = 0,1,2,\cdots$.
II. *There exists* $E^* \geq 0$ *such that* $\lim_{m\to\infty} E(w^m) = E^*$.
III. $\lim_{m\to\infty} \left\|E_{w_j}(w^m)\right\| = 0$, $j = 1,2,\ldots,n$.

Furthermore, if Proposition 4 is also valid, then we have the following strong convergence:

IV. *There exists a point* $w^* \in \Phi_0$ *such that* $\lim_{m\to\infty} w^m = w^*$.

4. Experimental Results

In order to validate the theoretical results and the effectiveness of the proposed algorithm (BGSENR), we have carried out some numerical experiments using four examples to verify the performance of the algorithm. We will then analyze the influence of parameters on the performance of the algorithm. The numerical experiments are used for a parity problem

two nonlinear function approximation problems, and the last numerical experiment is used the data sets to verify the classification performance of the network.

4.1. Example 1

To compare the classification performance of the BGL_1, BGL_2, BGENR, and BGSENR algorithms. We chose the data set from the UCI machine learning repository, which is publicly available set at https://archive.ics.uci.edu/ml/datasets.php. The data set is randomly split into training samples and testing samples, with percentages of 70% and 30%, respectively. The specifics of the data sets are presented below. Sonar is a classification issue that is linearly inseparable. The target of this data has been used to classify reflected sonar signals into metal chambers and shakes). Analyzing the performance of 208 input vectors, each of which has 60 interval components, which are used to test the presented training algorithms. Informatics indexes for preparation and testing are randomly selected from 208 information vectors, and the neurons of the organization contain 60 information units, 6 summation units and 1 production unit.

Using the parameters listed in Table 1, the operation was carried out in this example. According to Table 3, of the four algorithms, BGSENR performs the best in terms of testing accuracy, and since the training procedure is speedier, it enables faster convergence. The simulation results in this illustration are consistent with our theoretical analysis.

4.2. Example 2

A test for classification in a K-dimensional space with 2^K samples is the parity problem. The parity problem is also known as a normative classification problem. In order to match the exercise process and meet the loan from this process as shown in Table 1 and Table 2, we carefully choose the initial coefficients in this issue while taking into account the output layer, summation layer, and input layer.

We can see that the proposed learning algorithm (BGSENR), when compared to BGL_1, BGL_2, and BGENR, has the best learning accuracy, as predicted by Theorem 1. For each learning method, the average number of neurons eliminated through pruning over 10 trials for error and gradient norms is shown in Table 4. The error function of BGSENR has a monotonic decrease, and its gradient norm moves to zero more quickly than it does for BGL_2, and BGL_1 (see Figs. 2 and 3). The conclusion of this study is supported by this result. Additionally, Table 4 displays the average number of neurons removed per experiment and the duration of 10 experiments.

4.3. Example 3

To test the approximation ability of the proposed new algorithm, the following nonlinear function is used $cos(x), x \in [-1, 1]$, and the numbers of the training samples are 101 data. In this example, the structure, learning rate, regularization parameters and the training periods are given in Tables 1.

Table 5 displays the approximation preform of error training and testing, number of neurons eliminated, and running time of 10 experiments. From Figs. 4 – 7 results show that BGSENR has a better mapping capability than BGENR, BGL_1, and BGL_2, with the error decreasing monotonically as learning progresses and its approximation preform is better than the other. Table 5 demonstrates that the early results are very promising and that BGSENR has better speedup and generalization abilities than BGENR, BGL_1, and BGL_2.

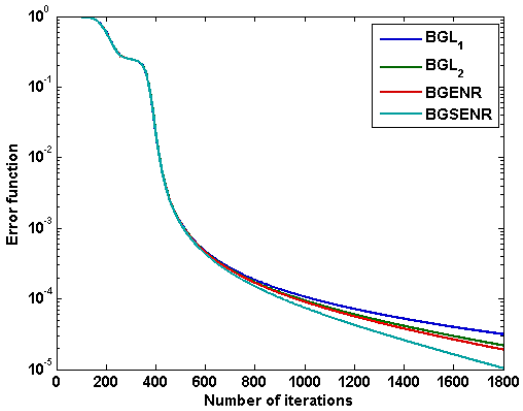

Fig. 2. The curve of error function for example 1.

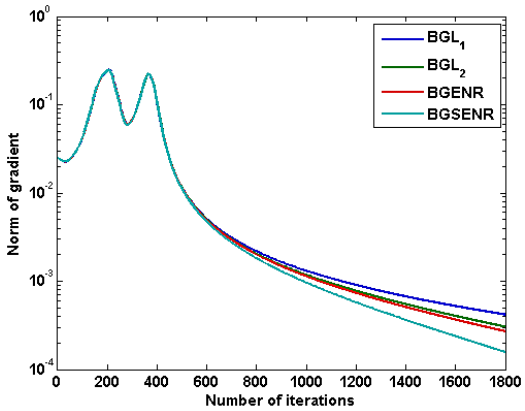

Fig. 3. The curve of norm of gradient for example 1.

Table 1. The learning parameters.

Problems	Network structure	Weight size	Max iteration	LeRe	RePa
Example 1	$5-4-1$	$[-0.5, 0.5]$	1800	0.03	0.0001/0.003
Example 2	$2-2-1$	$[-0.2, 0.2]$	10,000	0.05	0.001/0.003
Example 3	$2-4-1$	$[-0.5, 0.5]$	5,000	0.06	0.002/0.004
Example 4	$61-5-1$	$[-0.5, 0.5]$	5,000	0.6	0.0001/0.0005

Table 2. Samples of $4-$Bit parity problem.

Input				Output	Input				Output
1	1	1	1	0	-1	1	-1	-1	1
-1	1	1	1	1	1	1	-1	-1	0
1	1	1	-1	1	1	1	1	1	1
-1	1	1	-1	0	-1	1	1	1	0
-1	-1	-1	1	1	1	1	-1	1	0
1	-1	1	-1	0	-1	-1	1	-1	1
-1	-1	-1	-1	0	1	1	-1	1	1
1	-1	-1	-1	1	-1	1	-1	1	0

Table 3. Comparison results of classification accuracy for Example 4.

Learning methods	Accuracy training (%)	Accuracy testing (%)	Training time (s)
BGL_1	95.80	92.22	15.358937
BGL_2	96.40	93.27	15.398114
BGENR	94.72	90.98	15.849914
BGSENR	98.41	97.05	15.277646

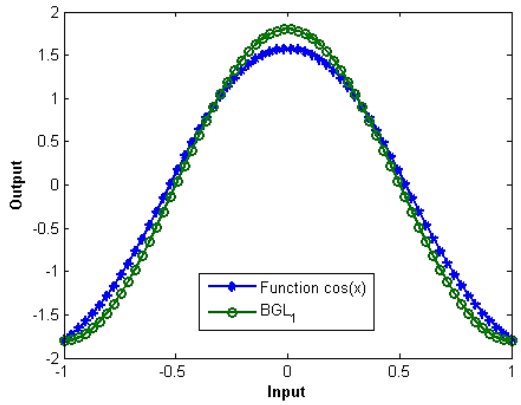

Fig. 4. The approximation preform by BGL_1 for example 2.

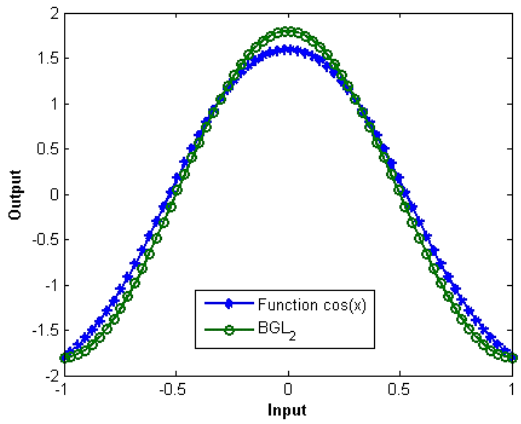

Fig. 5. The approximation preform by BGL_2 for example 2.

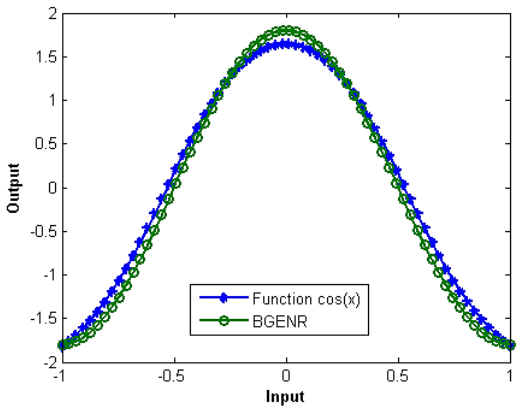

Fig. 6. The approximation preform by BGENR for example 2.

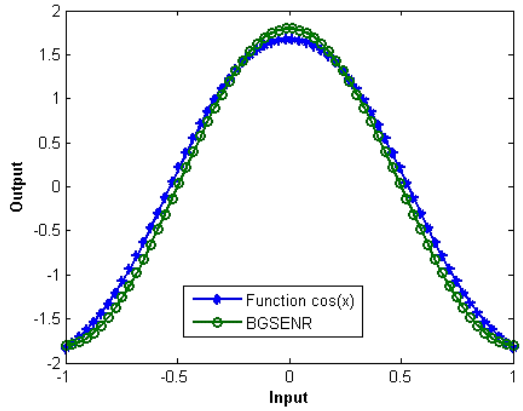

Fig. 7. The approximation preform by BGSENR for example 2.

4.4. Example 4

We use the 2D Gabor function to illustrate the possibility of BGSENR. As shown in Fig. 8, the two-dimensional Gabor function has the following form:

$$h(a,b) = \frac{1}{2\pi(0.5)} exp\left(\frac{a^2+b^2}{2(0.5)^2}\right) cos[2\pi(a+b)].$$

Then, 325 entry points were selected for training and testing to display the ability of the BGSENR. For training, 36 entry points were randomly selected from 6×6 enablers evenly spaced at -0.5×0.5 and -0.5×0.5 out of these 325 points. The remaining 289 points were randomly selected from 17×17 equally spaced at $-0.5 \leq a \leq 0.5$ and $-0.5 \leq b \leq 0.5$ for testing from these 325 points.

Table 6 compares the outcomes of the BGENR, BGL_2, BGL_1, and BGSENR algorithms using the typical number of neurons eliminated over 10 distinct iterations. The results show that our method performs well and has good planning capabilities for accelerated learning. As shown in Figs. 9 to 12, the proposed BGSENR algorithm generalizes stellar and achieves a smoother insertion during the training process than the BGENR, BGL_1, and BGL_2.

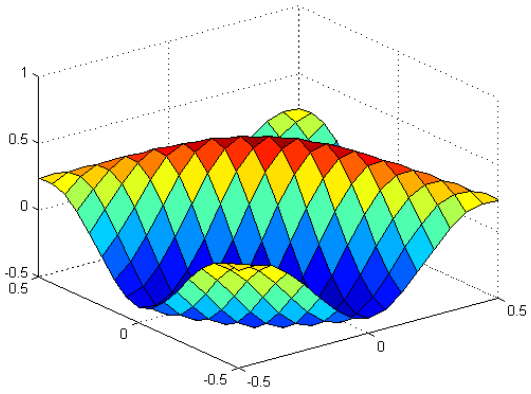

Fig. 8. Gabor function for example 3.

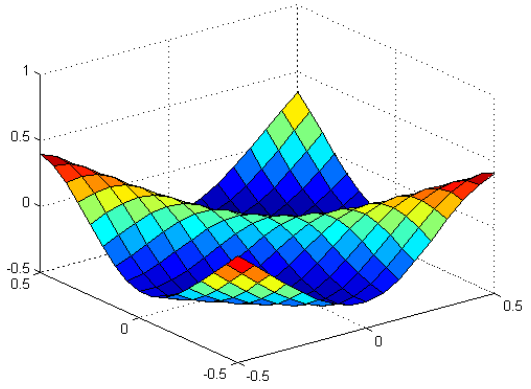

Fig. 9. The approximation preform by BGL_1 for example 3.

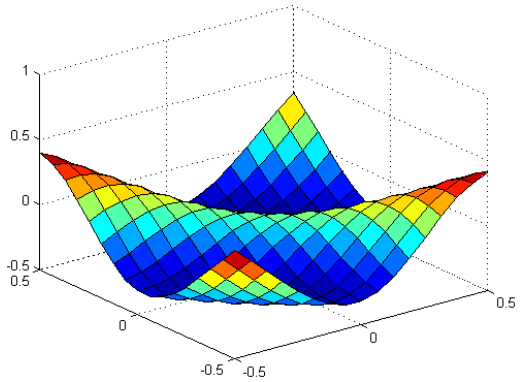

Fig. 10. The approximation preform by BGL_2 for example 3.

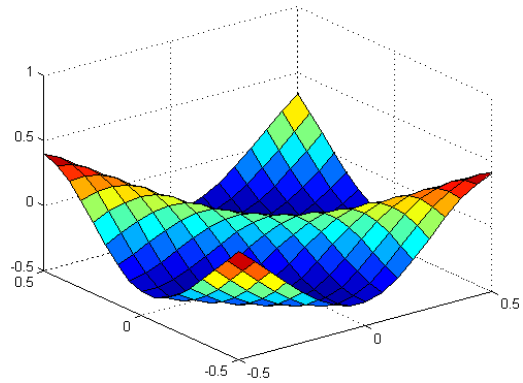

Fig. 11. The approximation preform by BGENR for example 3.

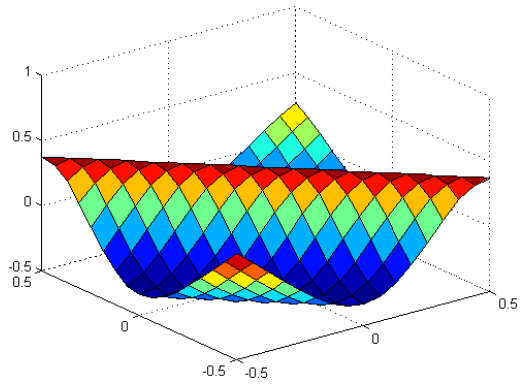

Fig. 12. The approximation preform by BGSENR for example 3.

4.5. Example 4

To compare the classification performance of the BGL_1, BGL_2, BGENR, and BGSENR algorithms. We chose the data set from the UCI machine learning repository, which is publicly available set at https://archive.ics.uci.edu/ml/datasets.php. The data set is randomly split into training samples and testing samples, with percentages of 70 % and 30 %, respectively. The specifics of the data sets are

presented below. Sonar is a classification issue that is linearly inseparable. The target of this data has been used to classify reflected sonar signals into metal chambers and shakes). Analyzing the performance of 208 input vectors, each of which has 60 interval components, which are used to test the presented training algorithms. Informatics indexes for preparation and testing are randomly selected from 208 information vectors, and the neurons of the organization contain 60 information units, 6 summation units and 1 production unit.

In this example, the operation was done using the parameters in Table 1. From Table 4, we found that our proposed method gives better accuracy (%), as the result in Table 4 was taken out of the ten best results obtained under the same conditions for four algorithms. The result of this comparison is generally acceptable to support theoretical funding.

Table 4. Numerical results for Example 1.

Learning methods	Average training error	Average testing error	Average number of neurons eliminated	Training time (s)
BGL_1	3.1433×10^{-5}	2.5336×10^{-4}	4.3	5.331998
BGL_2	2.1846×10^{-5}	1.8055×10^{-4}	4.1	5.356528
BGENR	1.9049×10^{-5}	1.5900×10^{-4}	3.9	5.360701
BGSENR	1.0383×10^{-5}	2.8449×10^{-5}	4.8	5.308263

Table 5. Numerical results for Example 2.

Learning methods	Average training error	Average testing error	Average number of neurons eliminated	Training time (s)
BGL_1	1.6039×10^{-5}	0.0048	5.9	0.555564
BGL_2	5.4765×10^{-5}	0.0044	6.1	0.566383
BGENR	4.1395×10^{-4}	0.0037	6.0	0.550159
BGSENR	8.8711×10^{-4}	0.0035	7.3	0.542739

Table 6. Numerical results for Example 3.

Learning methods	Average training error	Average testing error	Average number of neurons eliminated	Training time (s)
BGL_1	0.0407	0.0483	2.0	5.428512
BGL_2	0.0409	0.0480	2.2	4.493166
BGENR	0.0406	0.0481	2.5	5.483647
BGSENR	0.0101	0.0111	3.3	5.415376

5. Proofs

The following tow lemmas are a crucial tool for our analysis. For sake of convenience, we introduce the notations:

$$\Theta^{m+1,l} = \prod_{j=1}^{n}(w_j^{m+1}\cdot x^l). \quad (9)$$

Lemma 1. *Let the propositions 1 and 2 are satisfied, and that* $\{w^m\}_{m=0}^{\infty}$ *is grated by Equation (8). We have*

$$(\Theta^{m+1,l}-\Theta^{m,l})^2 \le C_1\sum_{j=1}^{n}\left(\Delta w_j^m\right)^2 \quad (10)$$

and

$$\sum_{l=1}^{L}\delta_l(\Pi^{m,l})\left[\Theta^{m+1,l}-\Theta^{m,l}\right] \le \left[-\frac{1}{\eta}+C_2\right]\sum_{j=1}^{n}\left(\Delta w_j^m\right)^2. \quad (11)$$

Proof. We write Following the Taylor expansion first and second orders. We write

$$\Theta^{m+1,l}-\Theta^{m,l} = \sum_{j=1}^{n}\left(\prod_{\substack{k=1\\k\neq j}}^{n}t_k\right)\left(\Delta w_j^m\cdot x^l\right) \quad (12)$$

and

$$\begin{aligned}\Theta^{m+1,l}-\Theta^{m,l} &= \sum_{j=1}^{n}\prod_{\substack{k=1\\k\neq i}}^{n}(w_k^m\cdot x^l)\left(\Delta w_j^m\cdot x^l\right)\\ &+\frac{1}{2}\sum_{\substack{j=1\\j\neq s}}^{n}\left[\prod_{\substack{k=1\\k\neq j\\k\neq s}}^{n}\left(t_k^{j,s}\right)\right]\left(\Delta w_j^m\cdot x^l\right)\left(\Delta w_s^m\cdot x^l\right)\\ &= \sum_{j=1}^{n}\prod_{\substack{k=1\\k\neq i}}^{n}(w_k^m\cdot x^l)\left(\Delta w_j^m\cdot x^l\right)+\sigma, \end{aligned} \quad (13)$$

where t_k and $t_k^{j,s}$ are in between $\left(w_j^{m+1}\cdot x^l\right)$ and $\left(w_j^m\cdot x^l\right)$. By using Proposition 2, easy to see that

$$|t_k|\le 2C_0,\ \ \left|t_k^{i,s}\right|\le 2C_0, \qquad k=1,2,\dots,n. \quad (14)$$

Thus, can be easily to obtained Equations (10), from (12) and (14). Through Equation (14), we write

$$
\begin{aligned}
|\sigma| &= \tfrac{1}{2}\sum_{j=1}^{n}[\prod_{\substack{k=1\\k\neq j}}^{n}(t_k^{j,s})]\,(\Delta w_j^m \cdot x^l)(\Delta w_s^m \cdot x^l) \\
&\leq \tfrac{1}{4}(2C_0)^{n-2}\max_{1\leq l\leq L}\|x^l\|^2\sum_{\substack{s=1\\j\neq s}}^{n}(\|\Delta w_j^m\|^2 + \\
&\qquad \|\Delta w_s^m\|^2) \\
&\leq C_2'\sum_{j=1}^{n}(\Delta w_j^m)^2, \qquad (15)
\end{aligned}
$$

where $C_2' = \frac{1}{4}(2C_0)^{n-2}\,\underset{1\leq l\leq L}{max}\|x^l\|^2$.

From collecting Equations (8), (13), (15) and Proposition 1. We have

$$
\begin{aligned}
&\textstyle\sum_{l=1}^{L}\delta_l\,(\Theta^{m,l})(\Theta^{m+1,l}-\Theta^{m,l}) \\
&= \textstyle\sum_{l=1}^{L}\delta_l(\Theta^{m,l})\,[\sum_{j=1}^{n}(\Delta w_j^m \cdot x^l)\prod_{\substack{k=1\\k\neq j}}^{n}(w_k^m \cdot \\
&\qquad x^l)+\sigma] \\
&\leq \textstyle\sum_{j=1}^{n}[\sum_{l=1}^{L}\delta_l(\Theta^{m,l})\prod_{\substack{k=1\\k\neq j}}^{n}(w_k^m \cdot x^l)\,x^l]\Delta w_j^m \\
&\quad + LC_0C_2'\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&= [-\tfrac{1}{\eta}+C_2]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2, \qquad (16)
\end{aligned}
$$

where $C_2 = LC_0C_2'$ This completes the proof.

Lemma 2. Let $\mathbb{Q}\colon \mathbb{R}^s \to \mathbb{R}$ is continuous and differentiable on a compact set $\tau \subset \mathbb{R}^s$ and that $\Omega = \left\{\phi \in \tau \middle| \frac{\partial F(\phi)}{\partial \phi} = 0\right\}$ has only finite number of point. If a sequence $\{\phi^z\}_{z=1}^{\infty} \in \tau$ satisfies

$$\lim_{z\to\infty}\|\phi^{z+1}-\phi^z\| = 0,\ \lim_{z\to\infty}\left\|\tfrac{\partial F(\phi^z)}{\partial \phi}\right\| = 0.$$

Then there exists a point $\phi^* \in \Omega$ such that

$$\lim_{z\to\infty}\phi^z = \phi^*.$$

Proof. is almost the same as Lemma 1 in [28] and the detail of the proof is omitted.

There are four parts to the proof of Theorem 1: statements (I), (II), (II), and (IV).

Proof to (*I*) of Theorem 1. Combining Equations (4), (10) (11), and the Taylor expansion, we get

$$
\begin{aligned}
&E(w^{m+1})-E(w^m) \\
&= \tfrac{1}{2}[\textstyle\sum_{l=1}^{L}(o^l-\Theta^{m+1,l})^2-\sum_{l=1}^{L}(o^l-\Theta^{m,l})^2] \\
&\quad +\lambda_1\textstyle\sum_{j=1}^{n}\left[\left(w_j^{m+1}\right)^2-\left(w_j^m\right)^2\right] \\
&\quad +\lambda_2\textstyle\sum_{j=1}^{n}\left[f\left(w_j^{m+1}\right)+f\left(w_j^m\right)\right] \\
&= \textstyle\sum_{l=1}^{L}\delta_l\,(\Theta^{m,l})(\Theta^{m+1,l}-\Theta^{m,l}) \\
&\quad +\tfrac{1}{2}\textstyle\sum_{l=1}^{L}\delta_l'\,(t_l')(\Theta^{m+1,l}-\Theta^{m,l})^2 \\
&\quad +\lambda_1\textstyle\sum_{j=1}^{n}\left[2w_j^m+\Delta w_j^m\right]\Delta w_j^m \\
&\quad +\lambda_2\textstyle\sum_{j=1}^{n}\left[f'\left(w_j^m\right)+f''(t_{j,m})\Delta w_j^m\right]\Delta w_j^m \\
&= \textstyle\sum_{j=1}^{n}[\sum_{l=1}^{L}\delta_l(\Theta^{m,l})\prod_{\substack{k=1\\k\neq j}}^{n}(w_k^m \cdot x^l)\,x^l]\Delta w_j^m \\
&\quad +\left[2\lambda_1 w_j^m+\lambda_2 f'(w_j^m)\right]\textstyle\sum_{j=1}^{n}\Delta w_j^m \\
&\quad +\tfrac{1}{2}\,LC_0C_1\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2+C_2\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&\quad +[\lambda_1+\lambda_2 f''(t_{j,m})]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&= \left[-\tfrac{1}{\eta}+\lambda_1+\lambda_2 f''(t_{j,m})\right]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&\quad +[\tfrac{1}{2}\,LC_0C_1+LC_0C_2']\textstyle\sum_{j=1}^{n}\|\Delta w_j^m\|^2 \\
&\leq -\left[\tfrac{1}{\eta}-\lambda_1-\lambda_2 f''\left(t_{j,m}\right)\right]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&\quad +[C_2+C_3]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2, \qquad (17)
\end{aligned}
$$

where $C_3 = \frac{1}{2}\,LC_0C_1$ t_l' is in between $\prod_{j=1}^{n}(w_j^{m+1}\cdot x^l)$ and $\prod_{j=1}^{n}(w_j^m \cdot x^l)$ and $t_{j,m}$ is in between w_j^{m+1} and w_j^m. With Equation (17), Proposition 3, and the Lagrangian Mean Value Theorem for $f(t)$ and Let $\mathbb{N} = f''(t_{j,m}) = \frac{3}{2a}$, then we have

$$
\begin{aligned}
&E(w^{m+1})-E(w^m) \\
&= [-\tfrac{1}{\eta}+\lambda_1+\lambda_2 f''(t_{j,m})+C_{2+}C_3]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&\leq -[\tfrac{1}{\eta}-\lambda_1-\lambda_2\mathbb{N}-C]\textstyle\sum_{j=1}^{n}(\Delta w_j^m)^2 \\
&\leq 0. \qquad (18)
\end{aligned}
$$

This completes the proof to statement (I) of Theorem 1.

Proof to (II) of the Theorem 1. From conclusion (I), we know that $E(w^m)$ is monotone. Additionally, it was bound below. Hence, there must exist $E^* \geq 0$ such that

$$\lim_{k\to\infty}E(w^m) = E^*.$$

The proof to (II) is thus completed.

Proof to (*III*) of the Theorem 1. From Equation (18) and Proposition 1 we can write $\alpha = \frac{1}{\eta}-\lambda_1-\mathbb{N}\lambda_2-C$ and set $\alpha > 0$. We have

$$
\begin{aligned}
E(w^{m+1}) &\leq E(w^m)-\alpha\textstyle\sum_{j=1}^{n}\left\|E_{w_j}(w^m)\right\|^2 \leq \cdots \\
&\leq E(w^0)-\alpha\textstyle\sum_{m=0}^{k}\sum_{j=1}^{n}\left\|E_{w_j}(w^m)\right\|^2.
\end{aligned}
$$

Since $E(w^{k+1}) > 0$ then we have

$$\alpha\textstyle\sum_{m=0}^{k}\sum_{j=1}^{n}\left\|E_{w_j}(w^m)\right\|^2 \leq E(w^0) < \infty.$$

Setting $k \to \infty$, we get

$$\textstyle\sum_{m=0}^{k}\sum_{j=1}^{n}\left\|E_{w_j}(w^m)\right\|^2 \leq \frac{1}{\alpha}\,E(w^0) < \infty.$$

Thus

$$\lim_{m\to\infty}\textstyle\sum_{j=1}^{n}\left\|E_{w_j}(w^m)\right\|^2 = 0.$$

Observation that

$$0 \leq \left\|E_{w_j}(w^m)\right\|^2 \leq \textstyle\sum_{j=1}^{n}\left\|E_{w_j}(w^m)\right\|^2 \to \infty.$$

Again setting $m \to \infty$, then we see that

$$\lim_{m\to\infty}\left\|E_{w_j}(w^m)\right\| = \lim_{m\to\infty}\left\|\Delta w_j^m\right\| = 0. \quad (19)$$

This completes the proof of (III).

Proof to (IV) of Theorem 1. Returning to the error function $E(w)$ defined in Equation (4) is a continuous and differentiable function. According to Lemma 2, Proposition 4, and Equation (19), we can easily get the desired result, that is, there is a point $w^* \in \varphi_0$ like so

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

This completes the proof to (IV).

6. Conclusions

Recently, in [29], an effective PSNN based on the online gradient method was provided with elastic net regularization or double regularization for pattern classification and function approximation. In order to indirectly incorporate the capabilities of higher-order networks while using fewer weights and processing units, this network uses product cells as the output units. The network has a regular structure, learns much more quickly, and can be made more complex by gradually adding additional units. The results of the simulation demonstrate accurate function approximation with good convergence properties.

In this study, the gradient algorithm's convergence results for PSNN with batch input training patterns are presented. The gradient algorithm is based on smoothing elastic net regularization into the error function (BGSENR). Additionally, the simulation outcomes for the four proposed numerical examples confirm our theoretical conclusions. We provide a convergence result for the BGSENR using smoothing elastic net regularization, which demonstrates that the error function decreases monotonically throughout training and that the network weights converge deterministically to the set of zeros of the gradient of the error function. As a result, it is anticipated that this new term will be able to train and optimize a wide variety of networks in addition to being able to solve a variety of challenging problems.

References

[1]. S. Das, J. Nayak, S. Nayak, S. Dey, Prediction of Life Insurance Premium during Pre-and Post-Covid-19: A Higher-Order Neural Network Approach, *Journal of The Institution of Engineers (India): Series B*, 2022, pp. 1-27.

[2]. S. Xu, Adaptive higher order neural network models and their applications in business, in Artificial higher order neural networks for economics and business, *IGI Global,* 2009, pp. 314-329.

[3]. C. L. Dunis, J. Laws, G. Sermpinis, Higher order and recurrent neural architectures for trading the EUR/USD exchange rate, *Quantitative Finance*, 11, 4, 2011, pp. 615-629.

[4]. Y. Shin, J. Ghosh, The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation, in *Proceedings of the* IEEE *International Joint Conference on Neural Networks (IJCNN-91)*, Seattle, Vol. 1, 1991, pp. 13-18.

[5]. E. Bisong, Batch vs. online learning, *Building Machine Learning and Deep Learning Models on Google Cloud Platform,* Apress, Berkeley, CA, 2019, pp. 199-201.

[6]. Y. Liu, D. Yang, N. Nan, L. Guo, J. Zhang, Strong convergence analysis of batch gradient-based learning algorithm for training pi-sigma network based on TSK fuzzy models, *Neural Processing Letters,* 43, No. 3, 2016, pp. 745-758.

[7]. J. Sensors Nayak, B. Naik, H. S. Behera, A hybrid PSO-GA based Pi sigma neural network (PSNN) with standard back propagation gradient descent learning for classification, in *Proceedings of the International IEEE Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT' 2014)*, 2014, pp. 878-885.

[8]. E. Egrioglu, U. Yolcu, E. Bas, Intuitionistic high-order fuzzy time series forecasting method based on pi-sigma artificial neural networks trained by artificial bee colony, *Granular Computing*, 4, 4, 2019, pp. 639-654.

[9]. H. Swapna Rekha, J. Nayak, H. S. Behera, Pi-sigma neural network: Survey of a decade progress, *Computational Intelligence in Pattern Recognition,* Springer, Singapore, 2020, pp. 429-441.

[10]. E. Bas, E. Egrioglu, O. Karahasan, A Pi-Sigma artificial neural network based on sine cosine optimization algorithm, *Granular Computing*, 2021, pp. 1-8.

[11]. O. Yılmaz, E. Bas, E. Egrioglu, The training of Pi-Sigma artificial neural networks with differential evolution algorithm for forecasting, *Computational Economics,* 59, No. 4, 2022, pp. 1699-1711.

[12]. N. Panda, S. K. Majhi, Improved spotted hyena optimizer with space transformational search for training pi-sigma higher order neural network, *Computational Intelligence,* 36, No. 1, 2020, pp. 320-350.

[13]. H. Zou, T. Hastie, Regularization and variable selection via the elastic net, *Journal of the royal statistical society: series B (statistical methodology),* 67, 2, 2005, pp. 301-320.

[14]. W. Shen, J. Wang, S. Ma, Doubly regularized portfolio with risk minimization, in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, June 2014.

[15]. Y. Li, S. Wang, P. X. K. Song, N. Wang, L. Zhou, J. Zhu, Doubly regularized estimation and selection in linear mixed-effects models for high-dimensional longitudinal data, *Statistics and its Interface*, 11, 4, 2018, pp. 721.

[16]. A. Zaib, T. Ballal, S. Khattak, T. Y. Al-Naffouri, A Doubly Regularized Linear Discriminant Analysis Classifier with Automatic Parameter Selection, *IEEE Access,* 9, 2021, pp. 51343-51354.

[17]. Y. Liu, D. Yang, C. Zhang, Relaxed conditions for convergence analysis of online back-propagation algorithm with L2 regularizer for Sigma-Pi-Sigma neural network, *Neurocomputing*, 272, 2018, pp. 163-169.

[18]. K. Nakamura, B. W. Hong, Adaptive weight decay for deep neural networks, *IEEE Access,* 7, 2019, pp. 118857-118865.
[19]. T. Van Laarhoven, L2 regularization versus batch and weight normalization, *arXiv preprint arXiv: 1706.05350*.
[20]. A. Cortiella, K. C. Park, A. Doostan, Sparse identification of nonlinear dynamical systems via reweighted ℓ1-regularized least squares, *Computer Methods in Applied Mechanics and Engineering,* 376, 2021, pp. 113620.
[21]. X. Qian, H. Huang, X. Chen, T. Huang, Efficient construction of sparse radial basis function neural networks using L1-regularization, *Neural Networks,* 94, 2017, pp. 239-254.
[22]. K. P. Burnham, D. R Anderson, Multimodel inference: understanding AIC and BIC in model selection, *Sociological Methods & Research,* 33, No. 2, 2004, pp. 261-304.
[23]. Z. Xu, H. Zhang, Y. Wang, X. Chang, Y. Liang, L1/2 regularization, *Science China Information Sciences* 53, no. 6 , 2010, pp. 1159-1169.
[24]. K. S. Mohamed, W. Wu, Y. Liu, A modified higher-order feed forward neural network with smoothing regularization, *Neural Network World,* 27, No. 6, 2017, pp. 577-592.
[25]. Y. Liu, Z. Li, D. Yang, K. S. Mohamed, J. Wang, W. Wu, Convergence of batch gradient learning algorithm with smoothing L1/2 regularization for Sigma–Pi–Sigma neural networks, *Neurocomputing,* 151, 2015, pp. 333-341.
[26]. K. S. Mohamed, Batch Gradient Learning Algorithm with Smoothing L 1 Regularization for Feedforward Neural Networks, *Computers*, 12, 1, 2022, p. 4.
[27]. Q Fan, J. M. Zurada, W. Wu, Convergence of online gradient method for feedforward neural networks with smoothing L1/2 regularization penalty, *Neurocomputing*, 131, 2014, pp. 208-216.
[28]. W. Wu, L. Li, J. Yang, Y. Liu, A modified gradient-based neuro-fuzzy learning algorithm and its convergence, *Information Sciences*, 180, 9, 2010, pp. 1630-1642.
[29]. K. S. Mohamed, A. A. O. Osman, K. Makin, M. N. A. Rabih, D. S. M. Suhail, Training pi-sigma neural network using double regularization, *Far East Journal of Electronics and Communications,* 26, 2022, pp. 1-16.