computers

Article

Batch Gradient Learning Algorithm with Smoothing L_1 Regularization for Feedforward Neural Networks

Khidir Shaib Mohamed [1,2]

[1] Department of Mathematics, College of Sciences and Arts in Uglat Asugour, Qassim University, Buraydah 51452, Saudi Arabia; k.idris@qu.edu.sa or khshm7@yahoo.com
[2] Department of Mathematics and Computer, College of Science, Dalanj University, Dilling P.O. Box 14, Sudan

Abstract: Regularization techniques are critical in the development of machine learning models. Complex models, such as neural networks, are particularly prone to overfitting and to performing poorly on the training data. L_1 regularization is the most extreme way to enforce sparsity, but, regrettably, it does not result in an NP-hard problem due to the non-differentiability of the 1-norm. However, the L_1 regularization term achieved convergence speed and efficiency optimization solution through a proximal method. In this paper, we propose a batch gradient learning algorithm with smoothing L_1 regularization (BGSL_1) for learning and pruning a feedforward neural network with hidden nodes. To achieve our study purpose, we propose a smoothing (differentiable) function in order to address the non-differentiability of L_1 regularization at the origin, make the convergence speed faster, improve the network structure ability, and build stronger mapping. Under this condition, the strong and weak convergence theorems are provided. We used N-dimensional parity problems and function approximation problems in our experiments. Preliminary findings indicate that the BGSL_1 has convergence faster and good generalization abilities when compared with BG$L_{1/2}$, BGL_1, BGL_2, and BGS$L_{1/2}$. As a result, we demonstrate that the error function decreases monotonically and that the norm of the gradient of the error function approaches zero, thereby validating the theoretical finding and the supremacy of the suggested technique.

Keywords: convergence; batch gradient learning algorithm; feedforward neural networks; smoothing L_1 regularization

Citation: Mohamed, K.S. Batch Gradient Learning Algorithm with Smoothing L_1 Regularization for Feedforward Neural Networks. *Computers* **2023**, *12*, 4. https://doi.org/10.3390/computers12010004

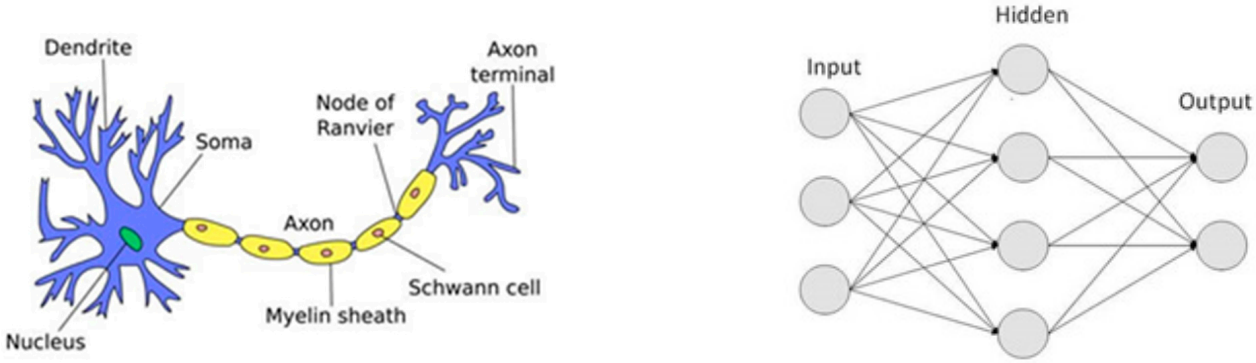
Academic Editor: Paolo Bellavista

Received: 6 December 2022
Revised: 13 December 2022
Accepted: 16 December 2022
Published: 23 December 2022

1. Introduction

Artificial neural networks (ANNs) are computational networks based on biological neural networks. These networks form the basis of the human brain's structure. Similar to neurons in a human brain, ANNs also have neurons that are interconnected to one another through a variety of layers. These neurons are known as nodes. The human brain is made up of 86 billion nerve cells known as neurons. They are linked to 1000 s of other cells by axons. Dendrites recognize stimulation from the external environment as well as inputs from sensory organs. These inputs generate electric impulses that travel quickly through the neural network. A neuron can then forward the message to another neuron to address the issue or not forward it at all. ANNs are made up of multiple nodes that mimic biological neurons in the human brain (See Figure 1). A feedforward neural network (FFNN) is the first and simplest type of ANN, and now it contributes significantly and directly to our daily lives in a variety of fields, such as education tools, health conditions, economics, sports, and chemical engineering [1–5].

The most widely used learning strategy in FFNNs is the backpropagation method [6]. There are two methods for training the weights: batch and online [7,8]. In the batch method, the weights are modified after each training pattern is presented to the network, whereas in the online method, the error is accumulated during an epoch and the weights are modified after the entire training set is presented.



(**a**) (**b**)

Figure 1. (**a**) The biological neural networks (**b**) The artificial neural network structure.

Overfitting in mathematical modeling is the creation of an analysis that is precisely tailored to a specific set of data and, thus, may fail to fit additional data or predict future findings accurately [9,10]. An overfitted model is one that includes more parameters than can be justified by the data [11]. Several techniques, such as cross-validation [12], early stopping [13], dropout [14], regularization [15], big data analysis [16], or Bayesian regularization [17], are used to reduce the amount of overfitting.

Regularization methods are frequently used in the FFNN training procedure and have been shown to be effective in improving generalization performance and decreasing the magnitude of network weights [18–20]. A term proportional to the magnitude of the weight vector is one of the simplest regularization penalty terms added to the standard error function [21,22]. Many successful applications have used various regularization terms, such as weight decay [23], weight elimination [24], elastic net regularization [25], matrix regularization [26], and nuclear norm regularization [27].

Several regularization terms are made up of the weights, resulting in the following new error function:

$$E(W) = \check{E}(W) + \lambda \|W\|_q^q \tag{1}$$

where $\check{E}(w)$ is the standard error function depending on the weights w, λ is the regularization parameter, and $\|\cdot\|_q$ is the q-norm is given by

$$\|W\|_q = \left(|w_1|^q + |w_2|^q + \cdots + |w_N|^q\right)^{\frac{1}{q}}$$

where $(0 \leq q \leq 2)$. The gradient descent algorithm is a popular method for solving this type of problem (1). The graphics of the L_0, $L_{1/2}$, L_1, L_2, elastic net, and L_∞ regularizers in Figure 2 show the sparsity. The sparsity solution, as shown in Figure 2, is the first point at which the contours touch the constraint region, and this will coincide with a corner corresponding to a zero coefficient. It is obvious that the L_1 regularization solution occurs at a corner with a higher possibility, implying that it is sparser than the others. The goal of network training is to find W^* so that $E(W) = \min E(W)$. The weight vectors' corresponding iteration formula is

$$W^{new} = W - \eta \frac{dE(W)}{dW} \tag{2}$$

L_0 regularization has a wide range of applications in sparse optimization [28]. As the L_0 regularization technique is an NP-hard problem, optimization algorithms, such as the gradient method, cannot be immediately applied [29]. To address this issue, ref. [30] proposes smoothing L_0 regularization with the gradient method for training FFNN. According to the regularization concept, Lasso regression was proposed to obtain the sparse solution based on L_1 regularization to reduce the complexity of the mathematical model [31].

Lasso quickly evolved into a wide range of models due to its outstanding performance. To achieve maximally sparse networks with minimal performance degradation, neural networks with smoothed Lasso regularization were used [32]. Due to its oracle properties, sparsity, and unbiasedness, the $L_{1/2}$ regularizer has been widely utilized in various studies [33]. A novel method for forcing neural network weights to become sparser was developed by applying $L_{1/2}$ regularization to the error function [34]. L_2 regularization is one of the most common types of regularization since the 2-norm is differentiable and learning can be advanced using a gradient method [35–38]; with L_2 regularization, the weights provided are bounded [37,38]. As a result, L_2 regularization is useful for dealing with overfitting problems.

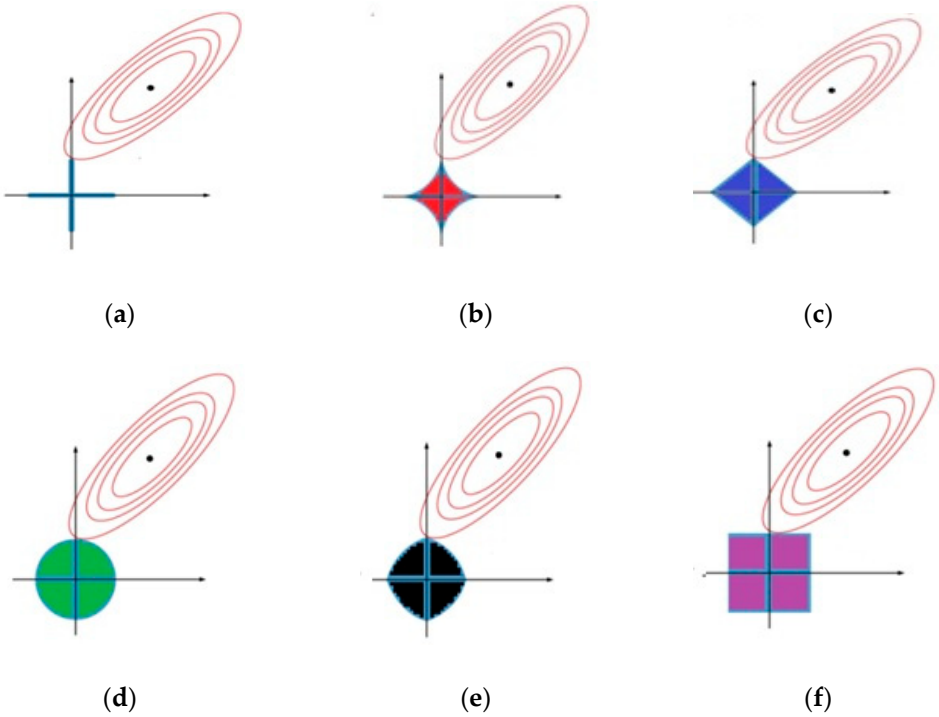


Figure 2. The sparsity property of different regularization (**a**) L_0 regularization, (**b**) $L_{1/2}$ regularization, (**c**) L_1 regularization, (**d**) L_2 regularization, (**e**) elastic net regularization, (**f**) L_∞ regularization.

The batch update rule, which modifies all weights in each estimation process, has become the most prevalent. As a result, in this article, we concentrated on the gradient method with a batch update rule and smoothing L_1 regularization for FFNN. We first show that if certain Propositions 1–3 are met, the error sequence will be uniformly monotonous, and the algorithm will be weakly convergent during training. Secondly, if there are no interior points in the error function, the algorithm with weak convergence is strongly convergent with the help of Proposition 4. Furthermore, numerical experiments demonstrate that our proposed algorithm eradicates oscillation and increases the computation learning algorithm better than the standard $L_{1/2}$ regularization, L_1 regularization, L_2 regularization, and even the smoothing $L_{1/2}$ regularization methods.

The following is the rest of this paper. Section 2 discusses FFNN, the batch gradient method with L_1 regularization (BGL_1), and the batch gradient method with smoothing L_1 regularization (BGSL_1). The materials and methods are given in Section 3. Section 4 displays some numerical simulation results that back up the claims made in Section 3. Section 5 provides a brief conclusion. The proof of the convergence theorem is provided in "Appendix A".

2. Network Structure and Learning Algorithm Methodology

2.1. Network Structure

A three-layer neural network based on error back-propagation is presented. Consider a three-layer structures consisting of **N** input layers, **M** hidden layers and 1 output layer. Given that $\mathbf{g} : \mathbb{R} \to \mathbb{R}$ can be a transfer function for the hidden and output layers, this is typically, but not necessarily, a sigmoid function. Let $w_0 = (w_{01}, w_{02}, \ldots, w_{0M})^T \in \mathbb{R}^M$ be the weight vector between all the hidden layers and the output layer, and let denoted $w_j = (w_{j1}, w_{j2}, \ldots, w_{jN})^T \in \mathbb{R}^N$ be the weight vector between all the input layers and the hidden layer $j (j = 1, 2, \ldots, M)$. To classify the offer, we write all the weight parameters in a compact form, i.e., $W = (w_0^T, w_1^T, \ldots, w_M^T) \in \mathbb{R}^{M+NM}$, and we have also given a matrix $V = (w_1, w_2, \ldots, w_M)^T \in \mathbb{M}^{N \times M}$. Furthermore, we define a vector function

$$\boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{f}(\boldsymbol{x_1}), \boldsymbol{f}(\boldsymbol{x_2}), \ldots, \boldsymbol{f}(\boldsymbol{x_N}))^{\boldsymbol{T}} \tag{3}$$

where $x = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N$. For any given input $\xi \in \mathbb{R}^M$ the output of the hidden neuron is $F(V\,\xi)$, and the final output of the network is

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{w_0} \cdot \boldsymbol{F}(\boldsymbol{V}\,\boldsymbol{\xi})) \tag{4}$$

where $w_0 \cdot F(V\,\xi)$ represents the inner product between the vectors $\boldsymbol{w^0}$ and $F(V\,\xi)$.

2.2. Modified Error Function with Smoothing L_1 Regularization (BGSL$_1$)

Given that the training set is $\left\{\xi^l, O^l\right\}_{l=1}^{L} \subset \mathbb{R}^N \times \mathbb{R}$, where O^l is the desired ideal output for the input ξ^l. The standard error function $E(W)$ without regularization term as following

$$\check{E}(W) = \frac{1}{2}\sum_{l=1}^{L}\left[O^l - f\left(w_0 \cdot F\left(V\xi^l\right)\right)\right]^2 = \sum_{l=1}^{L} f_l\left(w_0 \cdot F\left(V\xi^l\right)\right) \tag{5}$$

where $f_l(t) := \frac{1}{2}\left[O^l - f_l(t)\right]^2$. Furthermore, the gradient of the error function is given by

$$\check{E}_{w_0}(W) = \sum_{l=1}^{L} f_l'\left(w_0 \cdot F\left(V\xi^l\right)\right) F\left(V\xi^l\right) \tag{6}$$

$$\check{E}_{w_j}(W) = \sum_{l=1}^{L} f_l'\left(w_0 \cdot F\left(V\xi^l\right)\right) w_{0l} f_l'\left(w_j \cdot \xi^l\right)\xi^l \tag{7}$$

The modified error function $E(W)$ with L_1 regularization is given by

$$E(W) = \check{E}(W) + \lambda \sum_{j=1}^{M} |w_j| \tag{8}$$

where $|W|$ denoted the absolute value of the weights. The purpose of the network training is to find W^* such that

$$E\,(W^*) = minE\,(W) \tag{9}$$

The gradient method is a popular solution for this type of problem. Since Equation (8) involves the absolute value, this is a combinatorial optimization problem, and the gradient method cannot be employed to immediately minimize such an optimization problem. However, in order to estimate the absolute value of the weights, we recognize the use of a continuous and differentiable function to replace L_1 regularization by smoothing in (8). The error function with smoothing L_1 regularization can then be adapted by

$$E(W) = \check{E}(W) + \lambda \sum_{j=1}^{M} h(w_j), \tag{10}$$

where $h(x)$ is any continuous and differentiable functions. Specifically, we use the following piecewise polynomial function as:

$$h(t) = \begin{cases} |t \mid if \ |t| \geq m \\ -\frac{1}{8m^3}t^4 + \frac{3}{4m}t^2 + \frac{3}{8}m \ , \ if \mid t| < m, \end{cases} \tag{11}$$

where m is a suitable constant. Then the gradient of the error function is given by

$$E_W(W) = \left(E^T_{w_0}(W),\ E^T_{w_1}(W)\ , E^T_{w_2}(W),\ \cdots,\ E^T_{w_M}(W)\right)^T \tag{12}$$

The gradient of the error function in (10) with respect to w_j is given by

$$E_{w_j}(W) = \check{E}_{w_j}(W) + \lambda h'\left(w_j\right) \tag{13}$$

The weights $\left\{W^k\right\}$ updated iteratively starting from an initial value W^0 by

$$W^{k+1} = W^k + \Delta W^k,\ k = 0, 1, 2, \cdots, \tag{14}$$

and

$$\Delta w^k_j = -\eta\left[\check{E}_{w^k_j}(W) + \lambda h'\left(w^k_j\right)\right] \tag{15}$$

where $\eta > 0$ is learning rate, and λ is regularization parameter.

3. Materials and Methods

It will be necessary to prove the convergence theorem using the propositions below.

Proposition 1. *$|f(t)|$, $|f\prime(t)|$, $|f''(t)|$ and $|F(t)|$, $|F\prime(t)|$ are uniformly bounded for $t \in \mathbb{R}$.*

Proposition 2. *$\|w^k_0\|$ $(k = 0, 1, \cdots)$ is uniformly bounded.*

Proposition 3. *η and λ are chosen to satisfy: $0 < \eta < 1/(\lambda\mathbb{A} - C_1)$, where*

$$\begin{aligned} &C_1 = L(1 + C_2)C_3 max\{C_2, C_5\} + \tfrac{1}{2}L(1 + C_2)C_3 + \tfrac{1}{2}LC^2_3C^2_4C_5, \\ &C_2 = max\left\{\sqrt{\mathbb{B}}C_3, (C_3C_4)^2\right\}, \\ &C_3 = max\left\{\sup_{t\in\mathbb{R}}|f(t)|, \sup_{t\in\mathbb{R}}|f'(t)|,\ \sup_{t\in\mathbb{R}}|f''(t)|,\ \sup_{t\in\mathbb{R},1\leq l\leq L}\left|f'_l(t)\right|,\ \sup_{t\in\mathbb{R},1\leq l\leq L}\left|f''_l(t)\right|\right\}, \\ &C_4 = \min_{1\leq l\leq L}\|\xi^l\|,\ C_5 = \sup_{k\in\mathbb{N}}\|w^k_0\|. \end{aligned} \tag{16}$$

Proposition 4. *There exists a closed bounded region Θ such that $\left\{W^k\right\} \subset \Theta$, and set $\Theta_0 = \{W \in \Theta :\ E_w(W) = 0\}$ contains only finite points.*

Remark 1. *Both the hidden layer and output layer have the same transfer function, is tansig$(\cdot)$. A uniformly bounded weight distribution is shown in Propositions 1 and 2. Thus, Proposition 4 is reasonable. The Equation (10) and Proposition 1, $|f_l(t)|$, $\left|f'_l(t)\right|$, $\left|f''_l(t)\right|$ are uniformly bounded for Proposition 3. Regarding Proposition 2, we would like to make the following observation. This paper focuses mainly on simulation problems with $f(t)$ being a sigmoid function satisfying Proposition 1. Typically, simulation problems require outputs of 0 and 1, or -1 and 1. To control the magnitude of the weights w_0, one can change the desired output into $0 + \alpha$ and $1 - \alpha$, or $-1 + \alpha$ and $1 - \alpha$, respectively, where $\alpha > 0$ is a small constant. Actually, a more important reason of doing*

so is to prevent the overtraining, cf. *[39]. In the case of sigmoid functions, when $f(t)$ is bounded, the weights for the output layer are bounded.*

Theorem 1. *Let the weight $\left\{W^k\right\}$ be generated by the iteration algorithm (14) for an arbitrary initial value W^0, the error function $E(W)$ be defined by (10) and if propositions 1–3 are valid, then we have*

I. $E\left(W^{k+1}\right) \leq E\left(W^k\right),\ k = 0, 1, \cdots;$

II. *There exists $E^* \geq 0$ such that* $\lim_{k\to\infty} E_{w_j}\left(W^k\right) = E^*.$

III. $\lim_{k\to\infty} \|\Delta W^k\| = 0,$

IV. *Further, if proposition 4 is also valid, we have the following strong convergence*

V. *There exists a point $W^* \in \Theta_0$ such that* $\lim_{k\to\infty} W^k = W^*.$

Note: *It is shown in conclusion (I) and (II) that the error function sequence $\left\{E\left(W^k\right)\right\}$ is monotonic and has a limit (II). According to conclusions (II) and (IV), $E\left(W^k\right)$ and $E_W\left(W^k\right)$ are weakly converging. The strong convergence of $\{W^k\}$ is mentioned in Conclusion (V).*

We used the following strategy as a neuron selection criterion by simply computing the norm of the overall outgoing weights from the neuron number to ascertain whether a neuron number in the hidden units will survive or be removed after training. There is no standard threshold value in the literature for eliminating redundant weighted connections and redundant neurons from the initially assumed structure of neural networks. The sparsity of the learning algorithm was measured using the number of weights with absolute values of $\leq$0.0099 and $\leq$0.01, respectively, according to ref. [40]. In this study, we chose 0.00099 as a threshold value at random, which is less than the existing thresholds in the literature. This procedure is repeated ten times. Algorithm 1 describes the experiment procedure.

Algorithm 1 The learning algorithm	
Input	Input the dimension M, the number N of the nodes, the number maximum iteration number K, the learning rate η, the regularization parameter λ, and the sample training set is $\left\{\xi^l, O^l\right\}_{l=1}^{L} \subset \mathbb{R}^N \times \mathbb{R}$.
Initialization	Initialize randomly the initial weight vectors $w_0^0 = (w_{0,1}^0, \cdots, w_{0,M}^0)^T \in \mathbb{R}^M$ and $w_j^0 = (w_{j0}^0, \cdots, w_{jN}^0)^T \in \mathbb{R}^{\mathbf{N}}$ $j(j = 1, 2, \ldots, M)$
Training	For $k = 1, 2, \cdots, K$ do Compute the error function Equation (10). Compute the gradients Equation (15). Update the weights w_0^0 and w_j^0 $(1 \leq j \leq M)$ by using Equation (14). end
Output	Output the final weight vectors w_0^K and w_j^K $(1 \leq j \leq M)$

4. Experimental Results

The simulation results for evaluating the performance of the proposed BGSL_1 algorithm are presented in this section. We will compare BGSL_1 performance to that of four common regularization algorithms: the batch gradient method with $L_{1/2}$ regularization (BG$L_{1/2}$), the batch gradient method with smoothing $L_{1/2}$ regularization (BGS$L_{1/2}$), the batch gradient method with L_1 regularization (BGL_1), and the batch gradient method with L_2 regularization (BGL_2). Numerical experiments on the N-dimensional parity and function approximation problems support our theoretical conclusion.

4.1. N-Dimensional Parity Problems

The N-dimensional parity problem is another popular task that generates a lot of debate. If the input pattern contains an odd number of ones, the output criterion is one; alternatively, the output necessity is zero. An N-M-1 architecture (N inputs, M hidden nodes, and 1 output) is employed to overcome the N-bit parity problem. The well-known XOR problem is simply a 2-bit parity problem [41]. Here, the 3-bit and 6-bit parity problems are used as an example to test the performance of BGSL_1. The network has three layers: input layers, hidden layers, and an output unit.

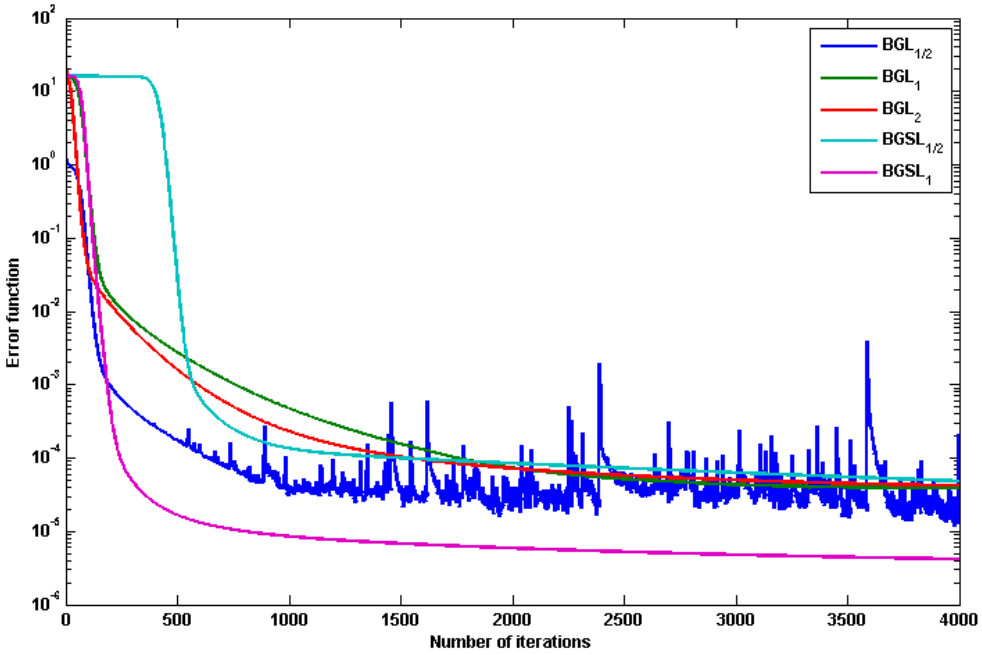
Table 1 shows the parameter settings for the corresponding network, where **LR** and **RP** are abbreviations for the learning rate and regularization parameter, respectively. Figures 3 and 4 show the performance results of BGL_2, BGL_1, BG$L_{1/2}$, BGS$L_{1/2}$, and BGSL_1 for 3-bit and 6-bit parity problems, respectively. As illustrated by Theorem 1, the error function decreases monotonically in Figures 3a and 4a, and the norm of the gradients of the error function approaches zero in Figures 3b and 4b. According to the comparison results, our proposal demonstrated superior learning ability and faster convergence. This corresponds to our theoretical analysis. Table 2 displays the average error and running time of ten experiments, demonstrating that BGSL_1 not only converges faster but also has better generalization ability than others.

Table 1. The learning parameters for parity problems.

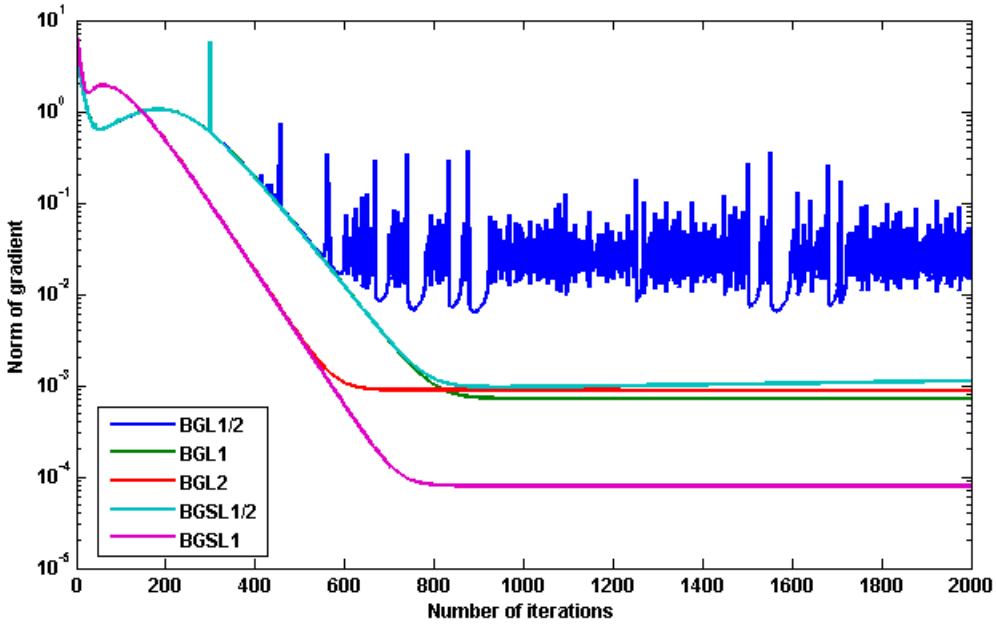
Problems	Network Structure	Weight Size	Max Iteration	LR	RP
3-bit parity	3-6-1	$[-0.5, 0.5]$	2000	0.009	0.0003
6-bit parity	6-20-1	$[-0.5, 0.5]$	3000	0.006	0.003

Table 2. Numerical results for parity problems.

Problems	Learning Algorithms	Average Error	Norm of Gradient	Time (s)
3-bit parity	BG$L_{1/2}$	3.7979×10^{-7}	0.0422	1.156248
	BGL_1	5.4060×10^{-7}	7.1536×10^{-4}	1.216248
	BGL_2	9.7820×10^{-7}	8.7826×10^{-4}	1.164721
	BGS$L_{1/2}$	1.7951×10^{-8}	0.0011	1.155829
	BGSL_1	7.6653×10^{-9}	7.9579×10^{-5}	1.135742
6-bit parity	BG$L_{1/2}$	8.1281×10^{-5}	1.1669	52.225856
	BGL_1	3.8917×10^{-5}	0.0316	52.359129
	BGL_2	4.1744×10^{-5}	0.0167	52.196552
	BGS$L_{1/2}$	4.8349×10^{-5}	0.0088	52.210994
	BGSL_1	4.1656×10^{-6}	0.0015	52.106554

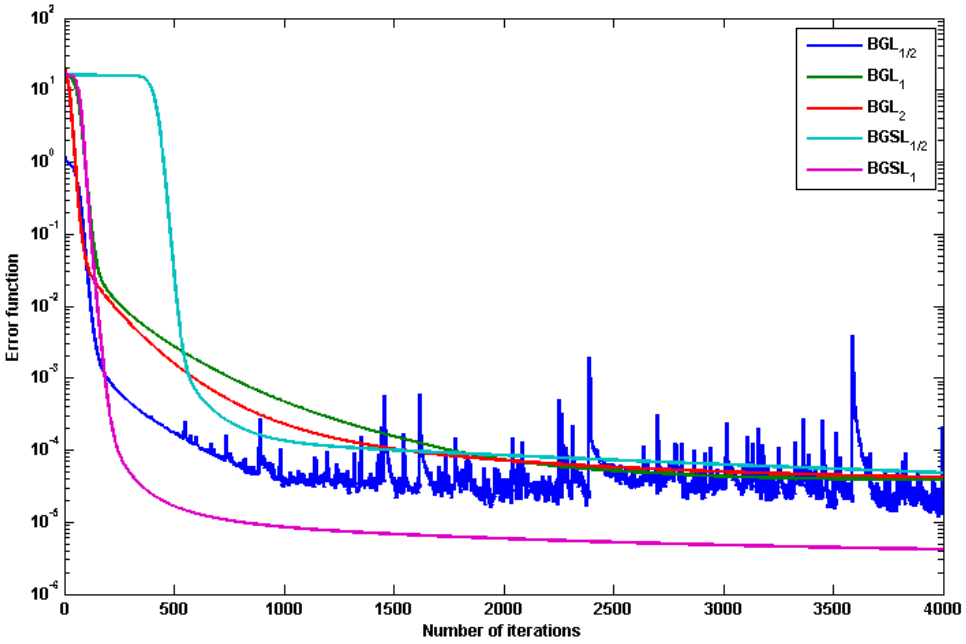


(**a**)

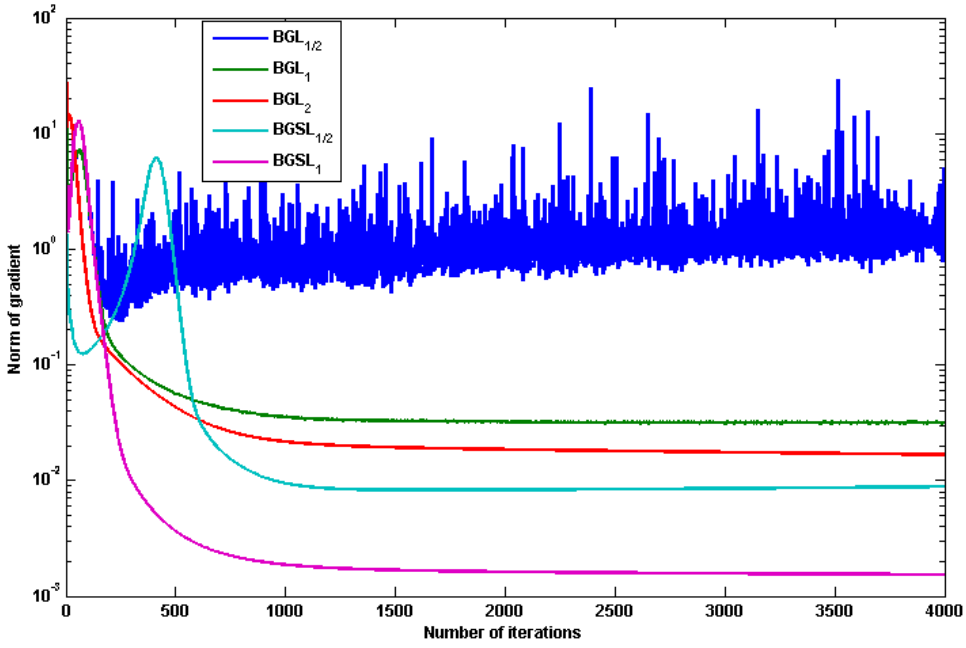


(**b**)

Figure 3. The performance results of five different algorithms based on 3-bit parity problem: (**a**) The curve of error function, (**b**) The curve of norm of gradient.



(**a**)



(**b**)

Figure 4. The performance results of five different algorithms based on 6-bit parity problem: (**a**) The curve of error function, (**b**) The curve of norm of gradient.

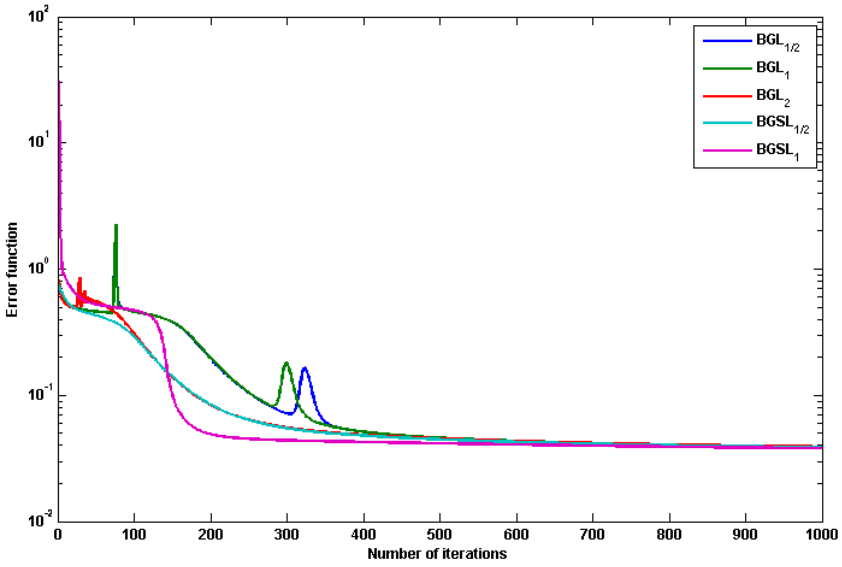
4.2. Function Approximation Problem

A nonlinear function has been devised to compare the approximation capabilities of the above algorithms:

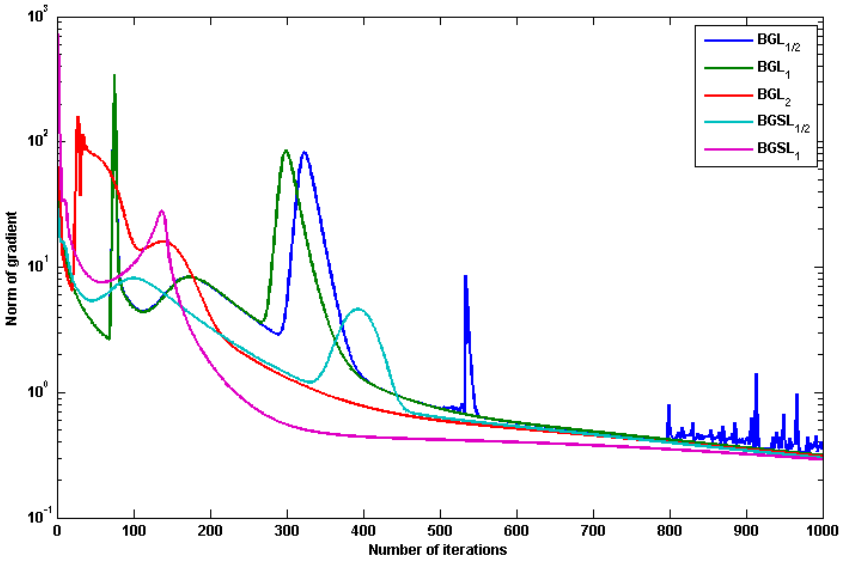
$$G(x) = \frac{1}{2}x - sin(x) \tag{17}$$

where $x \in [-4, 4]$ and chooses 101 training samples from an evenly spaced interval of $[-4, 4]$. The initial weight of the network is typically generated at random within a given interval; training begins with an initial point and gradually progresses to a minimum of error along the slope of the error function, which is chosen stochastically in $[-0.5, 0.5]$. The training parameters are as follows: 0.02 and 0.0005 represent the learning rate (η) and parameter regularization (λ), respectively. The stop criteria are set to 1000 training cycles.

The average error and norm of the gradient and running time of 10 experiments are presented in Table 3. Through the results obtained from Figure 5a,b, respectively, we see that BGSL_1 has a better mapping capability than BGS$L_{1/2}$, BG$L_{1/2}$, BGL_1 and BGL_2, with the error decreasing monotonically as learning proceeds and its gradient go to zero. Table 3 shows the preliminary results are extremely encouraging and that the speedup and generalization ability of BGSL_1 is better than BGS$L_{1/2}$, BG$L_{1/2}$, BGL_1 and BGL_2.



(**a**)



(**b**)

Figure 5. The performance results of five different algorithms based on function approximate problem: (**a**) The curve of error function, (**b**) The curve of norm of gradient.

Table 3. Numerical results for function approximation problem.

Learning Algorithms	Average Error	Norm of Gradient	Time (s)
BG$L_{1/2}$	0.0388	0.3533	4.415500
BGL_1	0.0389	0.3050	4.368372
BGL_2	0.0390	0.3087	4.368503
BGS$L_{1/2}$	0.0386	0.2999	4.349813
BGSL_1	0.0379	0.2919	4.320198

5. Discussion

Tables 2 and 3, respectively, show the performance comparison of the average error and the average norms of gradients under our five methods over the 10 trials. Table 2 shows the results of N-dimensional parity problems using the same parameters, while Table 3 shows the results of function approximation problems using the same parameters.

The comparison convincing in Tables 2 and 3 shows that the BGSL_1 is more efficient and has better sparsity-promoting properties than BG$L_{1/2}$, BGL_1, BGL_2 and even than the BGS$L_{1/2}$. In addition to that, Tables 2 and 3 show that our proposed algorithm is faster than that of all numerical results. $L_{1/2}$ regularization is sparser than the traditional L_1 regularization solution. Recently, in ref. [34], the BGS$L_{1/2}$ also shows that the sparsity is better than that of BG$L_{1/2}$. The results of ref. [33] show that the $L_{1/2}$ regularization has been demonstrated to have the following properties: unbiasedness, sparsity, and oracle properties.

We obtained all three numerical results for five different methods using one hidden layer of FFNNs. Our new method proposed a sparsification technique for FFNNs can be extended to encompass any number of hidden layers.

6. Conclusions

L_1 regularization is thought to be an excellent pruning method for neural networks. L_1 regularization, on the other hand, is also an NP-hard problem. In this paper, we propose BGSL_1, a batch gradient learning algorithm with smoothing L_1 regularization for training and pruning feedforward neural networks, and we approximate the L_1 regularization by smoothing function. We analyzed some weak and strong theoretical results under this condition, and the computational results validated the theoretical findings. The proposed algorithm has the potential to be extended to train neural networks. In the future, we will look at the case of the online gradient learning algorithm with a smoothing L_1 regularization term.

Funding: The researcher would like to thank the Deanship of Scientific Research, Qassim University for funding the publication of this project.

Data Availability Statement: All data has been presented in this paper.

Acknowledgments: The tresearcher would like to thank the referees for their careful reading and helpful comments.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

To prove the strong convergence, we will use the following result, which is basically the same as Lemma 3 in [38]. So its proof is omitted.

Lemma A1. *Let* $F : \Theta \subset \mathbb{R}^n \to \mathbb{R}$ *be continuous for a bounded closed region* Θ*, and* $\Theta_0 = \{z \in \Theta : F(z) = 0\}$*. The projection of* Θ_0 *on each coordinate axis does not contain any interior point. Let the sequence* $\left\{z^k\right\}$ *satisfy:*

(a) $\lim_{k\to\infty} F\left(z^k\right) = 0$;
(b) $\lim_{k\to\infty} \|z^{k+1} - z^k\| = 0$.

Then, there exists a unique $z^* \in \Theta_0$ *such that* $\lim_{k\to\infty} z^k = z^*$.

There are four statements in the proof of Theorem 1, each one is shown in (I)–(IV). We use the following notations for convenience:

$$\sigma^k = \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2 \tag{A1}$$

From error function (10), we can write

$$E\left(W^{k+1}\right) = \frac{1}{2}\sum_{l=1}^{L} f_l\left(w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right)\right) + \lambda\sum_{j=0}^{M} h(w_j^{k+1}) \tag{A2}$$

and

$$E\left(W^{k}\right) = \frac{1}{2}\sum_{l=1}^{L} f_l\left(w_0^{k}\cdot F(V^{k}\xi^l)\right) + \lambda\sum_{j=0}^{M} h(w_j^{k}) \tag{A3}$$

Proof to (I) of Theorem 1. Using the (A2), (A3), and Taylor expansion. We have

$$\begin{aligned}
&E\left(W^{k+1}\right) - E\left(W^{k}\right)\\
&= \sum_{l=1}^{L}\left[f_l\left(w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right)\right) - f_l\left(w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right)\right]\\
&+\lambda\sum_{j=0}^{M}[h(w_j^{k+1}) - h(w_j^{k})]\\
&= \sum_{j=1}^{L} f_l'\left(w_0^{n}\cdot F\left(V^{k}\xi^l\right)\right)\left[F\left(w_0^{n}\cdot\left(V^{k+1}\xi^l\right)\right) - F\left(w_0^{n}\cdot\left(V^{k}\xi^l\right)\right)\right]\\
&+\lambda\sum_{j=0}^{M}\left[h\prime(w_j^k) + \tfrac{1}{2}h''\left(t_{k,j}\right)\Delta w_j^k\right]\cdot\Delta w_j^k\\
&+\tfrac{1}{2}\sum_{l=1}^{L} f_l''(s_{k,l})\left[w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right) - w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right]^2\\
&= \sum_{l=1}^{J} f_l'\left(w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right)\left(F\left(V^{k}\xi^l\right)\right)\Delta w_0^k + \lambda h\prime\left(w_0^k\right)\cdot\Delta w_0^k\\
&+\sum_{j=1}^{L} f_l'\left(w_0^{n}\cdot F\left(V^{k}\xi^l\right)\right)\left[F\left(V^{k+1}\xi^l\right) - F\left(V^{k}\xi^l\right)\right]\cdot w_0^k\\
&+\lambda\sum_{j=1}^{M} h\prime(w_j^k)\cdot\Delta w_j^k + \tfrac{\lambda}{2}h''\left(t_{k,j}\right)\sum_{j=0}^{M}\left(\Delta w_j^k\right)^2\\
&+\sum_{j=1}^{L} f_l'\left(w_0^{n}\cdot F\left(V^{k}\xi^l\right)\right)\left[F\left(V^{k+1}\xi^l\right) - F\left(V^{k}\xi^l\right)\right]\cdot\Delta w_0^k\\
&+\tfrac{1}{2}\sum_{l=1}^{L} f_l''(s_{k,l})\left[w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right) - w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right]^2\\
&\le -\left(\tfrac{1}{\eta} - \tfrac{\lambda}{2}h''\left(t_{k,j}\right)\right)\sum_{j=0}^{M}\left(\Delta w_j^k\right)^2\\
&+\tfrac{1}{2}\sum_{l=1}^{L} f_l''(s_{k,l})\left[w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right) - w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right]^2\\
&+\sum_{l=1}^{L} f_l'\left(w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right)\left[F\left(V^{k+1}\xi^l\right) - F\left(V^{k}\xi^l\right)\right]\Delta w_0^n\\
&+\tfrac{1}{2}\sum_{j=1}^{M}\sum_{l=1}^{L} f_l'\left(w_0^{k}\cdot F\left(V^{k}\xi^l\right)\right)w_{0j}^k f_l''\left(t_{k,j,l}\right)\left(\Delta w_j^k\cdot\xi^l\right)^2
\end{aligned} \tag{A4}$$

where $t_{k,l} \in \mathbb{R}$ is between $w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right)$ and $w_0^{k}\cdot F\left(V^{k}\xi^l\right)$ and $t_{k,j,l} \in \mathbb{R}$ is between $w_j^{k+1}\cdot\xi^l$ and $w_j^{k}\cdot\xi^l$. From (16), Proposition 1 and the Lagrange mean value theorem. We have

$$\begin{aligned}
&\|F\left(V^{k+1}\xi^l\right) - F\left(V^k\xi^l\right)^2\| \\
&= \left\| \begin{pmatrix} f\left(w_1^{k+1}\cdot\xi^l\right) - f\left(w_1^k\cdot\xi^l\right) \\ \vdots \\ f\left(w_q^{k+1}\cdot\xi^l\right) - f\left(w_q^k\cdot\xi^l\right) \end{pmatrix} \right\|^2 \\
&= \left\| \begin{pmatrix} f'\left(\widetilde{t}_{1,j,l}\right)\left(w_1^{k+1} - w_1^k\right)\cdot\xi^l \\ \vdots \\ f'\left(\widetilde{t}_{i,j,l}\right)\left(w_q^{k+1} - w_q^k\right)\cdot\xi^l \end{pmatrix} \right\|^2 \\
&\leq C_2 \sum_{j=1}^{M} \left(\Delta w_j^k\right)^2
\end{aligned} \tag{A5}$$

and

$$\|F(x)\| \leq \sqrt{\mathbb{B}} \sum_{t\in\mathbb{R}} |f(t)| \leq C_2, \tag{A6}$$

where $\widetilde{t}_{k,j,l} \in \mathbb{R}$ $(1 \leq i \leq \mathbb{B})$ is between $w_j^{k+1}\cdot\xi^l$ and $w_j^k\cdot\xi^l$. According to Cauchy-Schwarz inequality, Proposition 1, (16), (A5) and (A6). We have

$$\begin{aligned}
&\left| \tfrac{1}{2} \sum_{l=1}^{L} f_l''\left(t_{k,l}\right)\left(w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right) - w_0^k\cdot F\left(V^k\xi^l\right)\right)^2 \right| \\
&\leq \tfrac{C_2}{2} \sum_{l=1}^{L} \left(w_0^{k+1}\cdot F\left(V^{k+1}\xi^l\right) - w_0^k\cdot F\left(V^k\xi^l\right)\right)^2 \\
&\leq \tfrac{C_2}{2} \sum_{l=1}^{L} 2\left(C_2^2\left(\Delta w_0^k\right) + C_5^2\left(F\left(V^{k+1}\xi^l\right) - F\left(V^k\xi^l\right)\right)\right)^2 \\
&\leq C_6 \sum_{l=1}^{L} \left(\left(\Delta w_0^k\right)^2 + C_2 \sum_{j=1}^{M} \left(\Delta w_j^k\right)^2\right) \\
&\leq \mathrm{L}C_6(1 + C_2) \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2 \\
&\leq C_7 \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2,
\end{aligned} \tag{A7}$$

where $C_6 = C_2\ max\{C_2^2, C_5^2\}$ and $C_7 = LC_6(1 + C_2)$. In the same way, we have

$$\begin{aligned}
&\left| \sum_{l=1}^{L} f_l'\left(w_0^k\cdot F\left(V^k\xi^l\right)\right)\left(F\left(V^{k+1}\xi^l\right) - F\left(V^k\xi^l\right)\right)\Delta w_0^k \right| \\
&\leq \tfrac{C_3}{2} \sum_{l=1}^{L} \left(\left(\Delta w_0^k\right)^2 + C_2 \sum_{j=1}^{M} \left(\Delta w_j^k\right)^2\right) \\
&\leq \tfrac{1}{2} LC_3(1 + C_2) \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2 \\
&\leq C_8 \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2,
\end{aligned} \tag{A8}$$

where $C_8 = \frac{1}{2} LC_3(1 + C_2)$. It follows from Propositions 1 and 2 that

$$\begin{aligned}
&\left| \tfrac{1}{2} \sum_{j=1}^{M} \sum_{l=1}^{J} f_l'\left(w_0^k\cdot F\left(V^k\xi^l\right)\right) w_{0j}^k f_j''\left(t_{k,j,l}\right)\left(\Delta w_j^k\cdot\xi^l\right)^2 \right| \\
&\leq \tfrac{1}{2} LC_3^2C_4^2C_5 \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2 \\
&\leq C_9 \sum_{j=0}^{M} \left(\Delta w_j^k\right)^2,
\end{aligned} \tag{A9}$$

where $C_9 = \frac{1}{2} LC_3^2C_4^2C_5$.

Let $C_1 = C_7 + C_8 + C_9$.

A combination of (A4) to (A9), and from $h(x)$ it easy to obtained $h(t) \in \left[\frac{3}{8}m, +\infty\right)$, $h\prime(t) \in [-1,\ 1]$, $h''(t) \in \left[0,\ \frac{3}{2m}\right]$, and $\overline{\mathbb{A}} = 3/2\omega$. We have

$$\begin{aligned}
E\left(W^{k+1}\right) - E\left(W^k\right) &\leq -\left[\tfrac{1}{\eta} - \tfrac{\lambda}{2} h\prime\left(t_{k,j}\right)\right] \sum_{j=1}^{M} \left(\Delta w_j^k\right)^2 + C_1 \sum_{j=1}^{M} \left(\Delta w_j^k\right)^2 \\
&\leq -\left[\tfrac{1}{\eta} - \tfrac{\lambda}{2}\overline{\mathbb{A}} - C_1\right] \sum_{j=1}^{M} \left(\Delta w_j^k\right)^2 \ \leq 0.
\end{aligned} \tag{A10}$$

Conclusion (I) of Theorem 1 is proved if the Proposition 3 is valid. □

Proof to (II) of Theorem 1. Since the nonnegative sequence $\left\{E\left(W^k\right)\right\}$ is monotone and bounded below, there must be a limit value $E^* < 0$ such that

$$\lim_{k\to\infty} E\left(W^k\right) = E^*.$$

So conclusion (II) is proved. □

Proof to (III) of Theorem 1. Proposition 1, (A10) and let $\mu > 0$. We have

$$\mu = \frac{1}{\eta} - \lambda\mathbb{A} - C_1 \tag{A11}$$

Thus, we can write

$$E\left(W^{k+1}\right) \leq E\left(W^k\right) - \mu\rho_n \leq \cdots \leq E\left(W^0\right) - \mu\sum_{q=0}^{n}\rho_n, \tag{A12}$$

when $E\left(W^{k+1}\right) > 0$ for any $k \geq 0$ and set $k \to \infty$, then we have

$$\sum_{q=0}^{\infty}\rho_n \leq \frac{1}{\mu}E\left(W^0\right) < \infty.$$

This with (12), (14) and (A1). We have

$$\lim_{k\to\infty}\|\Delta W^k\| = \lim_{k\to\infty}\|E_w\left(W^k\right)\| = 0 \tag{A13}$$

□

Proof to (IV) of Theorem 1. As a result, we can prove that the convergence is strong. Noting Conclusions (IV), we take $x = W$ and $h(x) = E_z(x)$. This together with the finiteness of Θ_0 (cf. Proposition 4), (A13), and Lemma 1 leads directly to conclusion (IV). This completes the proof. □

References

1. Deperlioglu, O.; Kose, U. An educational tool for artificial neural networks. *Comput. Electr. Eng.* **2011**, *37*, 392–402. [CrossRef]
2. Abu-Elanien, A.E.; Salama, M.M.A.; Ibrahim, M. Determination of transformer health condition using artificial neural networks. In Proceedings of the 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 15–18 June 2011; pp. 1–5.
3. Huang, W.; Lai, K.K.; Nakamori, Y.; Wang, S.; Yu, L. Neural networks in finance and economics forecasting. *Int. J. Inf. Technol. Decis. Mak.* **2007**, *6*, 113–140. [CrossRef]
4. Papic, C.; Sanders, R.H.; Naemi, R.; Elipot, M.; Andersen, J. Improving data acquisition speed and accuracy in sport using neural networks. *J. Sport. Sci.* **2021**, *39*, 513–522. [CrossRef]
5. Pirdashti, M.; Curteanu, S.; Kamangar, M.H.; Hassim, M.H.; Khatami, M.A. Artificial neural networks: Applications in chemical engineering. *Rev. Chem. Eng.* **2013**, *29*, 205–239. [CrossRef]
6. Li, J.; Cheng, J.H.; Shi, J.Y.; Huang, F. Brief introduction of back propagation (BP) neural network algorithm and its improvement. In *Advances in Computer Science and Information Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 553–558.
7. Hoi, S.C.; Sahoo, D.; Lu, J.; Zhao, P. Online learning: A comprehensive survey. *Neurocomputing* **2021**, *459*, 249–289. [CrossRef]
8. Fukumizu, K. Effect of batch learning in multilayer neural networks. *Gen* **1998**, *1*, 1E-03.
9. Hawkins, D.M. The problem of overfitting. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1–12. [CrossRef]
10. Dietterich, T. Overfitting and undercomputing in machine learning. *ACM Comput. Surv.* **1995**, *27*, 326–327. [CrossRef]
11. Everitt, B.S.; Skrondal, A. *The Cambridge Dictionary of Statistics*; Cambridge University Press: Cambridge, UK, 2010.
12. Moore, A.W. *Cross-Validation for Detecting and Preventing Overfitting*; School of Computer Science, Carnegie Mellon University: Pittsburgh, PA, USA, 2001.
13. Yao, Y.; Rosasco, L.; Caponnetto, A. On early stopping in gradient descent learning. *Constr. Approx.* **2007**, *26*, 289–315. [CrossRef]

14. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
15. Santos, C.F.G.D.; Papa, J.P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Comput. Surv.* **2022**, *54*, 1–25. [CrossRef]
16. Waseem, M.; Lin, Z.; Yang, L. Data-driven load forecasting of air conditioners for demand response using levenberg–marquardt algorithm-based ANN. *Big Data Cogn. Comput.* **2019**, *3*, 36. [CrossRef]
17. Waseem, M.; Lin, Z.; Liu, S.; Jinai, Z.; Rizwan, M.; Sajjad, I.A. Optimal BRA based electric demand prediction strategy considering instance-based learning of the forecast factors. *Int. Trans. Electr. Energy Syst.* **2021**, *31*, e12967. [CrossRef]
18. Alemu, H.Z.; Wu, W.; Zhao, J. Feedforward neural networks with a hidden layer regularization method. *Symmetry* **2018**, *10*, 525. [CrossRef]
19. Li, F.; Zurada, J.M.; Liu, Y.; Wu, W. Input layer regularization of multilayer feedforward neural networks. *IEEE Access* **2017**, *5*, 10979–10985. [CrossRef]
20. Mohamed, K.S.; Wu, W.; Liu, Y. A modified higher-order feed forward neural network with smoothing regularization. *Neural Netw. World* **2017**, *27*, 577–592. [CrossRef]
21. Reed, R. Pruning algorithms-a survey. *IEEE Trans. Neural Netw.* **1993**, *4*, 740–747. [CrossRef]
22. Setiono, R. A penalty-function approach for pruning feedforward neural networks. *Neural Comput.* **1997**, *9*, 185–204. [CrossRef] [PubMed]
23. Nakamura, K.; Hong, B.W. Adaptive weight decay for deep neural networks. *IEEE Access* **2019**, *7*, 118857–118865. [CrossRef]
24. Bosman, A.; Engelbrecht, A.; Helbig, M. Fitness landscape analysis of weight-elimination neural networks. *Neural Process. Lett.* **2018**, *48*, 353–373. [CrossRef]
25. Rosato, A.; Panella, M.; Andreotti, A.; Mohammed, O.A.; Araneo, R. Two-stage dynamic management in energy communities using a decision system based on elastic net regularization. *Appl. Energy* **2021**, *291*, 116852. [CrossRef]
26. Pan, C.; Ye, X.; Zhou, J.; Sun, Z. Matrix regularization-based method for large-scale inverse problem of force identification. *Mech. Syst. Signal Process.* **2020**, *140*, 106698. [CrossRef]
27. Liang, S.; Yin, M.; Huang, Y.; Dai, X.; Wang, Q. Nuclear norm regularized deep neural network for EEG-based emotion recognition. *Front. Psychol.* **2022**, *13*, 924793. [CrossRef]
28. Candes, E.J.; Tao, T. Decoding by linear programming. *IEEE Trans. Inf. Theory* **2005**, *51*, 4203–4215. [CrossRef]
29. Wang, Y.; Liu, P.; Li, Z.; Sun, T.; Yang, C.; Zheng, Q. Data regularization using Gaussian beams decomposition and sparse norms. *J. Inverse Ill Posed Probl.* **2013**, *21*, 1–23. [CrossRef]
30. Zhang, H.; Tang, Y. Online gradient method with smoothing $\ell 0$ regularization for feedforward neural networks. *Neurocomputing* **2017**, *224*, 1–8. [CrossRef]
31. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Methodol.* **1996**, *58*, 267–288. [CrossRef]
32. Koneru, B.N.G.; Vasudevan, V. Sparse artificial neural networks using a novel smoothed LASSO penalization. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *66*, 848–852. [CrossRef]
33. Xu, Z.; Zhang, H.; Wang, Y.; Chang, X.; Liang, Y. L1/2 regularization. *Sci. China Inf. Sci.* **2010**, *53*, 1159–1169. [CrossRef]
34. Wu, W.; Fan, Q.; Zurada, J.M.; Wang, J.; Yang, D.; Liu, Y. Batch gradient method with smoothing L1/2 regularization for training of feedforward neural networks. *Neural Netw.* **2014**, *50*, 72–78. [CrossRef] [PubMed]
35. Liu, Y.; Yang, D.; Zhang, C. Relaxed conditions for convergence analysis of online back-propagation algorithm with L2 regularizer for Sigma-Pi-Sigma neural network. *Neurocomputing* **2018**, *272*, 163–169. [CrossRef]
36. Mohamed, K.S.; Liu, Y.; Wu, W.; Alemu, H.Z. Batch gradient method for training of Pi-Sigma neural network with penalty. *Int. J. Artif. Intell. Appl. IJAIA* **2016**, *7*, 11–20. [CrossRef]
37. Zhang, H.; Wu, W.; Liu, F.; Yao, M. Boundedness and convergence of online gradient method with penalty for feedforward neural networks. *IEEE Trans. Neural Netw.* **2009**, *20*, 1050–1054. [CrossRef]
38. Zhang, H.; Wu, W.; Yao, M. Boundedness and convergence of batch back-propagation algorithm with penalty for feedforward neural networks. *Neurocomputing* **2012**, *89*, 141–146. [CrossRef]
39. Haykin, S. *Neural Networks: A Comprehensive Foundation*, 2nd ed.; Tsinghua University Press: Beijing, China; Prentice Hall: Hoboken, NJ, USA, 2001.
40. Liu, Y.; Wu, W.; Fan, Q.; Yang, D.; Wang, J. A modified gradient learning algorithm with smoothing L1/2 regularization for Takagi–Sugeno fuzzy models. *Neurocomputing* **2014**, *138*, 229–237. [CrossRef]
41. Iyoda, E.M.; Nobuhara, H.; Hirota, K. A solution for the n-bit parity problem using a single translated multiplicative neuron. *Neural Process. Lett.* **2003**, *18*, 233–238. [CrossRef]