Research Article

International Journal of Advanced Technology and Engineering Exploration, Vol 11(117)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2024.111100102

Convergence analysis of feedforward neural networks using the online gradient method with smoothing L_1 regularization

Khidir Shaib Mohamed[1, 2*] and Suhail Abdullah Alsaqer[3]
Department of Mathematics, College of Sciences, Qassim University, Saudi Arabia[1]
Department of Mathematics and Computer, College of Science, Dalanj University, Sudan[2]
Department of Computer Science, College of Computer, Qassim University, Saudi Arabia[3]

Received: 28-January-2024; Revised: 22-August-2024; Accepted: 25-August-2024

Abstract
An online gradient method is the simplest and most widely used training method for feed forward neural networks (FFNNs). However, a problem can arise with this method: sometimes the weights become very large, leading to overfitting. Regularization is a technique used to improve the generalization performance and prevent overfitting in networks. This paper focused on the convergence analysis of an online gradient method with L_1 regularization for training FFNNs. L_1 regularization promotes sparse models but complicates the convergence analysis process due to the inclusion of the absolute value function, which is not differentiable. To address this issue, an adaptive smoothing function was introduced into the error function to replace the L_1 regularization term at the origin. This approach encourages a sparser network structure by forcing weights to become smaller during training and eventually eliminating them after training. This strategy simplifies the network structure and accelerates convergence, as demonstrated by the numerical experiments presented in this paper. Additionally, it enables us to prove the convergence of the proposed training method. Numerical experiments based on 4-bit and 5-bit parity problems, Gaussian and hyperbolic function approximations, and Monk and Sonar classifications are provided to validate the theoretical findings and the superiority of the proposed algorithm.

Keywords
Online gradient method, Smoothing function, L1 regularization, Convergence, Numerical results, Feedforward neural network.

1.Introduction
One of the most often used architectures is feedforward neural networks (FFNNs), which has excellent representational capabilities, significant structural flexibility, and compatibility with a variety of training procedures. FFNNs are frequently employed in numerous applications [1–3]. For FFNNs, a prominent learning algorithm is backpropagation (BP) [4, 5]. The BP technique can be implemented in two workable ways: online updating and batch updating. Following the provision of all training samples, or the supply of each sample to the network during the learning process, they carry out the update. Two general approaches have been used to study the BP algorithm's convergence: the ordinary differential equation method [6] and the method based on the martingale convergence arguments [7].

*Author for correspondence

Overfitting is a major challenge for neural networks that controls their behavior and is considered a common and growing problem. An analysis that matches a specific set of data too closely or precisely is called overfitting, and as a result, it may not fit to other data sets or accurately forecast observations in the future [8]. An overfitted model is a mathematical model with more parameters than the data support [9]. However, there are many methods that have been used to get rid of this obstacle, for example: early stopping [10], dropout [11], noise injection [12], cross-validation [13], and others.

Regularization is one of the best and easiest techniques used to get rid of overfitting in general. L_1 norm and L_2 norm they are two commonly regularization techniques employed. A L_p regularization term (or L_p norm) is added to the error function as shown in Equation 1.

$$E(\mathcal{W}) = \tilde{E}(\mathcal{W}) + \lambda\|\mathcal{W}\|_p^p \quad (1)$$

where $\tilde{E}(\mathcal{W})$ is the typical error function depending on the weights $\mathcal{W}$, λ is the regularizer parameter, and $\|\cdot\|_{\mathcal{p}}$ is the $L_{\mathcal{p}}$ norm is given by

$$\|\mathcal{W}\|_{\mathcal{p}} = \sqrt[\mathcal{p}]{(|w_1|^{\mathcal{p}} + |w_2|^{\mathcal{p}} + \cdots + |w_M|^{\mathcal{p}})}$$

The general form of gradient learning algorithm for a training of error function in Equation 1 can be described as follows in Equation 2.

$$\mathcal{W}^{n+1} = \mathcal{W}^n - \eta_n E_{\mathcal{W}}(\mathcal{W}^n) \quad (2)$$

where $\mathcal{W}^n$, η_n and $E_{\mathcal{W}}(\mathcal{W}^n)$ denote the weight value, the learning rate at the n −th iteration and the gradient value at the point $\mathcal{W}^n$, respectively.

Through Equation 1, there are different forms of regulation depending on the value of $\mathcal{p}$. The weight decay regularization term, for $\mathcal{p} = 2$, can successfully regulate the increase in network weights [14–16]. However, it does not result in a sparse solution. Analytical solutions are available for the learning issue with weight decay regularization and the least squares loss function. Specifically, during training, the sum of the weights' norm is penalized using $L_{\mathcal{p}}$ regularization. A non-convex, non-smooth, global non-differentiable function is the $L_{\mathcal{p}}$ regularization ($0 < \mathcal{p} < 1$) [17, 18]. The two most widely used $L_{\mathcal{p}}$ regularizations are $L_{1/2}$ and L_1 regularizations [19–21]. They lessen the need for redundancy weights. By encouraging the surplus weights to take values near zero, the training process enhances generalization. However, the primary purpose of standard $L_{\mathcal{p}}$ regularization is to eliminate redundant weights; a node can only be eliminated if all of its outgoing weights have been nearly zero. L_1 regularization is typically applied to models in order to boost their sparsity [22, 23]. In other words, least absolute shrinkage and selection operator (LASSO) regression [24, 25] selects variables in addition to performing prediction by penalizing the L_1 norm, which tends to lower many coefficients to absolute zero. First presented for use in linear regression problems [26], Lasso is a novel strategy that reduces some coefficients and sets others to zero.

This paper aims to employ smoothing functions to approximate the L_1 regularization, drawing inspiration from the research, which utilized the batch gradient approach with smoothing L_1 regularization for FFNNs. As a result, it offers an online gradient method with smoothing L_1 regularization ($OGMSL_1$) for training FFNNs. It will demonstrate how applying smoothing L_1 regularization improves the gradient method's ability to differentiate between significant and insignificant weights. Under certain assumptions, we give the strong and weak convergence theorems for a novel algorithm. Numerical experiments also illustrate the theoretical results and the advantages of this algorithm.

Section 2 provides an overview of the literature. Section 3 explains the methods used. The results of the proposed method's convergence are presented in Section 4. It also covers the discussion on the findings. It is finally concluded in Section 5. To emphasize the key concepts of this study, the process for proving theorems is provided in *Appendix I.*

2.Literature review

Different types of regularizations have been applied in previous work, which do not apply to the current equation but have become commonly used to prevent overfitting. Examples include weight elimination [27], matrix regularization [28], and elastic net [29]. Thus proposed the elastic net, penalizing both the L_1 and L_2 norms with individual tuning parameters, as a way to achieve the best of both L_1 regularization method and L_2 regularization method. Elastic net regularization has attracted significant attention from researchers due to its effectiveness. A batch gradient technique with smoothing elastic net regularization is proposed in [30] for pruning feedforward polynomial neural networks, with a particular focus on pi-sigma neural networks. A more recent study, [31], focused on a novel Sigma-Pi-Sigma neural network that prevents overfitting by controlling network complexity through the use of batch gradient approach for L_1 and L_2 regularization. Additionally, a smoothing technique can be used to eliminate the numerical oscillation that is caused by the non-differentiability of L_1 plus L_2 regularization at the origin in order to approximate the objective function.

In the related work, smoothing $L_{\mathcal{p}}$ regularization ($0 \leq \mathcal{p} \leq 1$) is a target pruning and learning technique for machine learning and neural networks, and it has gained a lot of attention lately. The objective function of the normal $L_{\mathcal{p}}$ regularization ($0 \leq \mathcal{p} \leq 1$) is non-differentiable, so that the gradient method cannot be applied directly; instead, it can only be removed by using a smoothing technique to approximate the objective function. For pruning FFNNs, the online gradient technique with smoothing L_0 regularization is used [32], and the extreme learning machine's smoothing L_0 regularization is provided in [33]. For training pi-sigma neural networks, the online gradient method with smoothing L_0 regularization is used [34]. In [35], an entropy error function featuring smoothing L_0 regularization is suggested. Specifically, literature

[36–39] provides an excellent overview of the study of smoothing $L_{1/2}$ regularization in various types of networks, including BP FFNNs, sigma-pi-sigma order neural networks, Spiking FFNNs, and Takagi-Sugeno (T-S) fuzzy systems. Since the normal L_1 regularization has an objective function that is non-differentiable, and inspired by the work on smoothing L_0 and $L_{1/2}$ regularizations for FFNNs. Smoothing L_1 regularization has become a trend in many applications such as learning optimized structure of neural networks by hidden node pruning [40], sparse spatial filter using smooth L_1 regularization and a unique objective function reduction in [41], and a fast optimization method [42]. This study investigates an online gradient method with smoothing L1 regularization ($OGMSL_1$) for FFNNs, extending previous work [43] from the batch gradient method to the online gradient method.

This work is substantial as the performance and the theoretical analysis of online method are basically different from those of batch method.

3.Materials and methods

3.1The L_1 regularization method ($OGML_1$)

This study examin FFNNs, which has p input layers, q hidden layers, and 1 output layer. The weight vector connecting each hidden layer to the output layer is denoted by $w_0 = (w_{10}, w_{20}, \ldots, w_{q0})^{\mathrm{T}} \in \mathbb{R}^{\mathrm{q}}$. For $i = 1,2,\ldots,q$, the weight vector between each of the input layers and the hidden layer is represented by the symbol $w_{\mathrm{i}} = (w_{\mathrm{i1}}, w_{\mathrm{i2}}, \ldots, w_{\mathrm{ip}})^{\mathrm{T}} \in \mathbb{R}^{\mathrm{p}}$. In order to make things simpler, a matrix was provided $\mathcal{V} = (w_0^{\mathrm{T}}, w_0^{\mathrm{T}}, \ldots, w_{\mathrm{q}}^{\mathrm{T}}) \in \mathbb{R}^{\mathrm{q}\times\mathrm{p}}$ along with all of the weight parameters written in a compact form, or $\mathcal{W} = (w_1, w_2, \ldots, w_{\mathrm{q}})^{\mathrm{T}} \in \mathbb{R}^{\mathrm{q+pq}}$. Given that $\mathrm{G}: \mathbb{R}^{\mathrm{q}} \to \mathbb{R}^{\mathrm{q}}$ and $g: \mathbb{R} \to \mathbb{R}$ be activations for the hidden and output layers, respectively, this is usually a sigmoid function. Next, a vector function is defined in Equation 3.

$$G(x) = (g(x_1), g(x_2), \ldots, g(x_q))^T \quad (3)$$

When the weight $\mathcal{W}$ is learned, the actual output vector of the hidden neurons is $\mathrm{G}(\mathcal{V}\xi)$ for every input $\xi = (\xi_1, \xi_2, \ldots, \xi_q)^T \in \mathbb{R}^q$.

In terms of mathematics, the background determines the gradient (or derivative) of the cost function related to a certain condition with regard to the weights. To this end, let the input training samples be $\{\xi^j, O^j\}_{j=1}^J \subset \mathbb{R}^p \times \mathbb{R}$, where O^j is the input ξ^j is ideal desired output, and the neural network's true output is calculated by Equation 4.

$$y^j = g\left(w_0 \cdot G(\mathcal{V}\xi^j)\right) \quad (4)$$

where the inner product of the two vectors, w_0 and $\mathrm{G}(\mathcal{V}\xi^j)$, is represented by $w_0 \cdot \mathrm{G}(\mathcal{V}\xi^j)$. Then, the L_1 regularized error function ($OGML_1$), which is shown in Equation 5.

$$\boldsymbol{E(\mathcal{W}) = \frac{1}{2}\sum_{j=1}^{J}\left[O^j - g\left(w_0 \cdot G(\mathcal{V}\xi^j)\right)\right]^2 + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p}|w_{ik}| = \sum_{j=1}^{J} g_j\left(w_0 \cdot G(\mathcal{V}\xi^j)\right) + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p}|w_{ik}|} \quad (5)$$

where the regularization term's parameter is $\lambda > 0$, the absolute value is indicated by $|\cdot|$, the continuous $g_j(\mathrm{t})$ and its derivative are shown in Equation 6.

$$\boldsymbol{g_j(t) := \frac{1}{2}\left(O^j - g_j(t)\right)^2}$$

and

$$\boldsymbol{g_j'(t) := \left(O^j - g_j(t)\right)g_j(t), \quad t \in \mathbb{R}.}$$

Next, based on $\mathcal{W}$, the gradient error function is provided by Equations 6 and 7.

$$E_{w_{i0}}(\mathcal{W}) = \sum_{j=1}^{J} g_j'(w_0 \cdot G(\mathcal{V}\xi^j))\, g(w_i\xi^j) + \lambda sgn(w_{i0}) \quad (6)$$

$$E_{w_{ik}}(\mathcal{W}) = \sum_{j=1}^{J} g_j'(w_0 \cdot G(\mathcal{V}\xi^j))\, w_{i0} g'(w_i \cdot \xi^j)\xi_k^j + \lambda sgn(w_{ik}) \quad (7)$$

Given an initial weigths $\mathcal{W}^0 \in \mathbb{R}^{q+pq}$, $OGML_1$ then the weights $\{\mathcal{W}^{nJ+j}\}$ updated iteratively in Equation 8.

$$\mathcal{W}^{nJ+j} = \mathcal{W}^{nJ+j-1} - \eta_n \Delta_j^n \mathcal{W}^{nJ+j-1} \quad (8)$$

With Equations 9 and 10

$$\boldsymbol{\Delta_j^n w_{i0}^{nJ+j-1} = g_j'\left(w_0^{nJ+j-1} \cdot G(\mathcal{V}^{nJ+j-1}\xi^j)\right) \times g\left(w_i^{nJ+j-1}\xi^j\right) + \lambda sgn\left(w_{i0}^{nJ+j-1}\right)} \quad (9)$$

$$\boldsymbol{\Delta_j^n w_{ik}^{nJ+j-1} = g_j'\left(w_0^{nJ+j-1} \cdot G(\mathcal{V}^{nJ+j-1}\xi^j)\right) w_{i0}^{nJ+j-1} \times g'\left(w_i^{nJ+j-1}\xi^j\right)\xi_k^j + \lambda sgn\left(w_{ik}^{nJ+j-1}\right)} \quad (10)$$

where $k = 1,2,\cdots,p$; $j = 1,2,\cdots,J$; $n = 0,1,2,\cdots$, and λ is the regularization term parameter. In this case, η_n represents the learning rate in the n^{th} training period.

3.2The smoothing L_1 regularization method (OGMSL_1)

The connection weights are modified using the BP technique to accommodate for any errors discovered during learning. Since the error function (Equation 5) contains an absolute value, the online gradient method cannot be utilized instantly to it. A natural extension of well-known techniques such as sub-gradient descent to situations where the objective function is not differentiable everywhere is the sub-gradient approach. A workaround was proposed for this problem: the piecewise polynomial function $\boldsymbol{f(x)}$, which can roughly approximate $|\boldsymbol{x}|$. To begin with, create the smoothing function as shown in Equation 11.

$$f(x)=\begin{cases}|x|, & if\ |x|\geq\gamma\\ -\frac{x^4}{8\gamma^3}+\frac{3x^2}{4\gamma}+\frac{3\gamma}{8}, & if|x|<\gamma\end{cases} \quad (11)$$

In specifically, the piecewise polynomial function $f(x)$ is chosen, and the smoothing parameter $\gamma>0$ is used. To date, Equation 12 works.

$$f(t)\in\left[\frac{3}{8}\gamma,+\infty\right),\ f'(t)\in[-1,1],\ f''(t)\in\left[0,\frac{3}{2\gamma}\right] \quad (12)$$

Currently, the smoothing L_1 regularization error function (OGMSL_1) is Equation 13.

$$\boldsymbol{E(\mathcal{W})=\frac{1}{2}\sum_{j=1}^{J}\left[O^j-g\left(w_0\cdot G(\mathcal{V}\xi^j)\right)\right]^2}$$
$$\boldsymbol{+\lambda\sum_{i=1}^{q}\sum_{k=0}^{p}f(w_{ik})}$$
$$\boldsymbol{=\sum_{j=1}^{J}g_j(w_0\cdot G(\mathcal{V}\xi^j))+\lambda\sum_{i=1}^{q}\sum_{k=0}^{p}f(w_{ik})} \quad (13)$$

The gradient of the error function in the Equation 13 depending on $\mathcal{W}$, which is given by Equations 14 and 15.

$$E_{w_{i0}}(\mathcal{W})=\sum_{j=1}^{J}g_j'(w_0\cdot G(\mathcal{V}\xi^j))\,g(w_i\xi^j))+\lambda f'(w_{i0}) \quad (14)$$

$$E_{w_{ik}}(\mathcal{W})=\sum_{j=1}^{J}g_j'(w_0\cdot G(\mathcal{V}\xi^j))\,w_{i0}g'(w_i\cdot\xi^j)\xi_k^j+\lambda f'(w_{ik}) \quad (15)$$

Given an initial weight $\boldsymbol{\mathcal{W}^0}\in\mathbb{R}^{q+pq}$, OGMS$\mathbf{L_1}$ then the weights $\{\boldsymbol{\mathcal{W}^{nJ+j}}\}$ updated iteratively as shown in Equation 16

$$\mathcal{W}^{nJ+j}=\mathcal{W}^{nJ+j-1}-\eta_n\Delta_j^n\mathcal{W}^{nJ+j-1} \quad (16)$$

With Equations 17 and 18.

$$\boldsymbol{\Delta_j^n w_0^{nJ+j-1}=g_j'\left(w_0^{nJ+\tau-1}\cdot G(\mathcal{V}^{nJ+j-1}\xi^j)\right)}$$
$$\boldsymbol{\times g(w_i^{nJ+j-1}\xi^j)+\lambda f'(w_{i0}^{nJ+j-1})} \quad (17)$$

$$\boldsymbol{\Delta_j^n w_0^{nJ+j-1}=g_j'\left(w_0^{nJ+j-1}\cdot G(\mathcal{V}^{nJ+j-1}\xi^j)\right)}$$
$$\boldsymbol{w_{i0}^{nJ+j-1}\times g'(w_i^{nJ+j-1}\xi^j)\xi_k^j+\lambda f'(w_{ik}^{nJ+j-1})} \quad (18)$$

where $k=1,2,\cdots,p$; $j=1,2,\cdots,J$; $n=0,1,2,\cdots$, and λ is the regularization term parameter. In this case, η_n represents the learning rate in the n-th training period.

3.3Convergence results

The Euclidean norm of x is written as $\|x\|=\sqrt{\sum_{i=1}^{n}x_i^2}$ for $x=(x_1,x_2,\cdots,x_n)\in\mathbb{R}^n$. The stationary point set of the error function E($\mathcal{W}$) is represented by $\Theta_0=\{\mathcal{W}\in\Theta: E_w(\mathcal{W})=0\}$. The algorithm's convergence is predicated on the following premises:

Assumption 1. For $t\in\mathbb{R}$, $|g(t)|$ and $|g'(t)|$ are Lipchitz continuous;

Assumption 2. $0<\eta_n<1$, and $\sum_{n=0}^{\infty}\eta_n<\infty$.

Assumption 3. A bounded open set $\Theta\subset\mathbb{R}^n$ exists, such that $\mathcal{W}^n\subset\Theta$ $(n\in\mathbb{N})$.

Assumption 4. For each $s=1\cdots,pq+q$, there is no interior point in the set Θ_0.

Remark 1 *In order to ensure convergence, the learning rate η_n usually needs to meet certain assumptions $\sum_{n=0}^{\infty}\eta_n=\infty$ and $\sum_{n=0}^{\infty}\eta_n^2<\infty$ as in [44]. An ancillary assumption $\lim_{n\to\infty}\eta_n/\eta_{n+1}=1$ was discussed in [45]. A special condition which is basically $\eta_n=O(1/\eta_n)$ was desired in [46, 47]. As of now, the almost-cyclic learning result [48] still requires the condition $\eta_n=O(1/\eta_n)$. $\frac{1}{\eta_n}=\frac{1}{\eta_{n+1}}+\beta$, where $\beta>0$ is a constant, should be used to determine the learning rate for each cycle of the learning iteration (see [49], where an online gradient approach is explored). Recently, in [36] the learning rate η_n is selected to satisfy $0<\eta_n<1$, and $\sum_{n=0}^{\infty}\eta_n<\infty$ and provide more sparse solution and here η_n is small constant. This paper follows this condition.*

Theorem 1 *With an arbitrary initial value $\mathcal{W}^0$, let the weight $\{\mathcal{W}^{nJ+j}\}$ be generated by the iteration by Equation 16. The error function $E(\mathcal{W})$ is defined by Equation 13, if Assumptions 1 through 3 are true, the weight sequence $\{\mathcal{W}^{nJ+j}\}$ can be obtained that monotonically decreases, indicating that the weak convergence holds (Equation 19).*

$$\lim_{n\to\infty}\left\|E_w(\mathcal{W}^{nJ+j})\right\|=0 \quad (19)$$

Moreover, the following strong convergence was obtained if Assumption 4 is likewise true. $\mathcal{W}^*\in\Theta_0$ is a point that exists such that (Equation 20).

$$\lim_{n\to\infty} \mathcal{W}^{nJ+j} = \mathcal{W}^{*} \quad (20)$$

Remark 2 *Assumption 1 is satisfied by common activation functions like sigmoid functions, such as "is the tan-sigmoid function* $g(t) = tansig(t)$*," and includes the crucial situation that the network output is a nonlinear function. From a finite training set,* $\{\xi^j, O^j\}_{j=1}^{J} \subset \mathbb{R}^p \times \mathbb{R}$ *is randomly generated, where* $\xi^j \in \mathbb{R}^p$ *and* $O^j \in \mathbb{R}$*. Assumption 3 does not impose any restrictions on the condition* $\lambda \in (0,1)$ *because, in actuality, the penalty parameter* λ *should be small and positive constant. A common prerequisite for the learning rate* η *to ensure the convergence of an online gradient method is Assumption 4. A convergence theorem is demonstrated for the suggested learning approach. It is shown in Equation 51, and the proof is provided in the Appendix I.*

Remark 3 *Generally, the network's initial weight is generated at random intervals. The training process begins at an initial point and proceeds progressively to a minimum error along the error function's slope. This means that the direction of the network's convergence is established once the initial value is established. If a poor starting weight is chosen, the network's convergence direction may move in the direction of divergence, which results in the network's concussion not convergent. Furthermore, the choice of initial weight has an impact on the convergence rate of network training.*

Consequently, choosing a suitable starting weight appears to be crucial in order to quicken the network training process' convergence rate and prevent brain injuries during the research process. Through the use of a genetic algorithm, the neural network's initial weight can be optimized so that it jumps out of the local extremum, accelerating the BP network's convergence and increasing the network's convergence accuracy.

3.4Learning algorithm, data sets and model parameters

3.4.1Learnin algorithm

The online gradient method is used to examine one sample of data at a time, changing the model parameters after each sample. The online gradient technique is useful when the training data is too large to retain in memory or is being streamed in real-time. This is an example of the online gradient method used in algorithm 1. Remember that the online gradient can be applied with more complex models and cost functions; this is only a basic demonstration. The primary advantage of the live gradient method is that it does not necessitate loading all of the data into memory sequentially. The general steps of the suggested algorithm ($OGMSL_2$) are as follows:

Algorithm 1: Project for $OGMSL_1$

1.Initialize:	Initialize the model parameters, such as weights ($\mathcal{W}$), learning rate (η), regularization parameter (λ), and smoothing parameter (γ).
2.Receive Data:	Receive a new data point, such as a target label. This data point is often referred to as a "sample" or "example". Here is given the input training samples be $\{\xi^j, O^j\}_{j=1}^{J} \subset \mathbb{R}^p \times \mathbb{R}$, where ξ^j is the input O^j is ideal desired output.
3.Predict:	Use the current model parameters to make a prediction (output in Equation 4) on the new data point.
4.Calculate Error:	Calculate the error between the predicted output and the true target label in Equation 13
5.Calculate Gradient:	Calculate the gradient of the error function with respect to the all-model parameters.
6.Update Model:	Update the model parameters using the gradient and the learning rate.
7.Repeat:	Repeat steps 2-6 for each new data point received

3.4.2Datasets

In this study, two classification datasets, Monks and Sonar, (*Table 1*), were used to compare the accuracy of $OGMSL_1$ with other methods from University of California, Irvine (UCI) machine learning repository. The N-bit parity problem is a classic problem in machine learning, where the goal is to predict the parity (even or odd) of a N-bit binary vector. *Table 2* and *Table 3* shows the training samples for the 4-bit and 5-bit parity problems.

Table 1 Guidelines for datasets with classification issues

Problems	Data size	Input features	Classes	Dataset information links
MONK's	432	6	2	https://archive.ics.uci.edu/dataset/70
Sonar	208	60	2	https://archive.ics.uci.edu/dataset/151

Table 2 Four-dimensional parity problem samples

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 Five-dimensional parity problem samples

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

3.4.3Model parameters

The internal variables that the algorithm learns and modifies during the training process to create predictions or classifications are referred to as model parameters in machine learning models. These parameters may comprise coefficients or weights, contingent upon the particular model employed. In this case, tiny constants are used for the learning rate, regularization parameter, and smoothing parameter. *Table 4* shows the models parameters.

Table 4 Model parameters

Algorithms	η, λ, γ	Monks	Sonar	Gaussian function	Hyperbolic function	4-bit parity	5-bit parity
$OGML_{1/2}$	η	0.05	0.5	0.006	0.003	0.003	0.005
	λ	0.003	0.001	0.003	0.002	0.005	0.002
$OGML_1$	η	0.05	0.5	0.006	0.003	0.003	0.005
	λ	0.003	0.001	0.003	0.002	0.005	0.002
$OGML_2$	η	0.05	0.5	0.006	0.003	0.003	0.005
	λ	0.003	0.001	0.003	0.002	0.005	0.002
$OGML_{1/2}$	η	0.05	0.5	0.006	0.003	0.003	0.005
	λ	0.003	0.001	0.003	0.002	0.005	0.002
	γ	0.06	0.05	0.0055	0.005	0.005	0.009
$OGMSL_1$	η	0.05	0.5	0.006	0.003	0.003	0.005
	λ	0.003	0.001	0.003	0.002	0.005	0.002
	γ	0.56	0.56	0.0009	0.009	0.099	0.0009
Network structure		7−8−1	61−10−1	$2-2-1$	2–3–1	5–6–1	6–8–1
Weight Size		[–0.5,0.5]	[–0.5,0.5]	[−0.2,0.2]	[–0.2,0.2]	[–0.5,0.5]	[–0.5,0.5]
Max iteration		4000	5000	1500	2000	2500	3000

4.Results and discussion

This paper presents methods for analyzing numerical experiments, which are cost-effective calculations used to determine a model's response to variations in parameters. The algorithm $OGMSL_1$ is compared with other online gradient methods, including $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$, to demonstrate its sparsity and convergence speed. The comparison is made using five numerical problems: Monk problem, Sonar problem, Gaussian and Hyperbolic function approximation problems, 4-bit parity, and 5-bit parity problems.

4.1Benchmark classification problems

The benchmark datasets selected from the UCI Machine Learning Repository for simulation included both binary and multi-class classification problems. *Table 1* provides a complete list of the datasets, and *Table 4* shows the parameter setup for the corresponding networks. The data is split into the training and testing samples, and a predetermined percentage of 80% and 20%, respectively. For each dataset, the following normalization technique was adopted $x_{new} = 2(x - \bar{x})/(x_{max} - x_{min})$ to preprocess the training and testing samples. Four algorithms were used to assess the performance: tasks that required a significant amount of time to accomplish, training accuracy, and testing accuracy. As shown in *Table 5*, the proposed $OGMSL_1$ algorithm outperformed four other algorithms across all four metrics. $OGMSL_1$'s learning solutions for the classification tasks are more effective and better training/testing accuracy in terms of time, meaning that the exercise can be finished in the shortest amount of time.

4.2Higher-order gaussian function approximation problem

In this example, to verify the effectiveness of the proposed algorithm. The numerical experiment of Gaussian function problems is considered. By raising the content of the exponent to a power P, Equation 21 provides a more general formulation of a Gaussian function with a flat top and Gaussian falloff.

$$f(x) = A\ exp\left(\frac{-(x-x_0)^2}{2\sigma_x}\right)^p, x \in [-4,4] \quad (21)$$

Super-Gaussian functions, like the one shown here, are frequently employed in the formulation of Gaussian beams. In this case, the amplitude is represented by coefficient A, the center by x_0, and the blob's x spread by σ_x. The data in *Table 6* was supplemented with A = 0.05, $x_0 = 0.06$, and $\sigma_x = 0.03$ to construct the *Figures 1* and *2*. *Table 6* shows that the smoothing L_1 regularization has the best efficacy and least inaccuracy among all techniques. It has a stronger sparsity-promoting property and is better at drawing results, reducing time. The error function and gradient norm for L_1 regularization are monotone decreasing are monotone decreasing and more converge than the others. *Figures 1* and *2* shown that during the learning process, the error and the gradient's norm for the $L_{1/2}$ regularization oscillate. On the other hand, as Theorem 1 predicts, *Figures 1* and *2* show that the gradient norm and the error function for the smoothing L_1 regularization are monotone decreasing and exhibit better convergence than the others.

4.3Hyperbolic function approximation problem

Hyperbolic functions, arising from the application of sine and cosine functions to imaginary angles, are meromorphic in the complex plane and can be defined in various ways. In this instance, the approach involves approximating a specific function using a neural network (Equation 22).

$$f(x) = tanh(x),\ x \in [-4,4] \quad (22)$$

The aim of this test is to compare the behaviours of our $OGMSL_1$ with $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$. As expected by Theorem 1, *Figure 3* shows that the error functions of $OGMSL_1$ decline monotonically. Additionally, it is evident that $OGMSL_1$'s error and its gradients are fewer than those of $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$ when training with the same parameters. *Figure 4* illustrates how the norms of gradient of the error functions approach very small positive constants. *Table 7* also shows that our $OGMSL_1$ trains more quickly than the other five approaches, indicating that it is the quickest in actual use.

The aim of this test is to compare the behaviors of $OGMSL_1$ with $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$. As expected by Theorem 1, *Figure 3* shows that the error functions of $OGMSL_1$ decline monotonically. Additionally, it is evident that $OGMSL_1$'s error and its gradients are lower than those of $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$ when training with the same parameters. *Figure 4* illustrates how the norms of the gradients of the error functions approach very small positive constants. *Table 7* also shows that $OGMSL_1$ trains more quickly than the other five approaches, indicating that it is the fastest in actual use.

4.4Higher-dimensional parity problems

A constraint in sieve theory known as the parity problem in number theory hinders sieves from

providing accurate estimates in a wide range of prime-counting tasks. One challenging categorization problem is the N-dimensional parity problem. There are 2^N input patterns in the N −dimensional vector space. The 2-dimensional parity problem is simply the famous XOR problem. *Figures 5* to *8* display the values of the error function and the error function gradient for the XOR and parity problems, respectively. *Figures 5* and *7* demonstrate the monotonic reduction of the $OGMSL_1$ error functions. As predicted by Theorem 1, *Figures 6* and *8* show that the error function's gradient norms approach extremely small positive constants. This illustrates how the $OGMSL_1$ algorithm efficiently reduces the weight magnitudes. Additionally, *Table 8* demonstrates that the $OGMSL_1$ trains faster than the others. Theorem 1 is also supported by these findings.

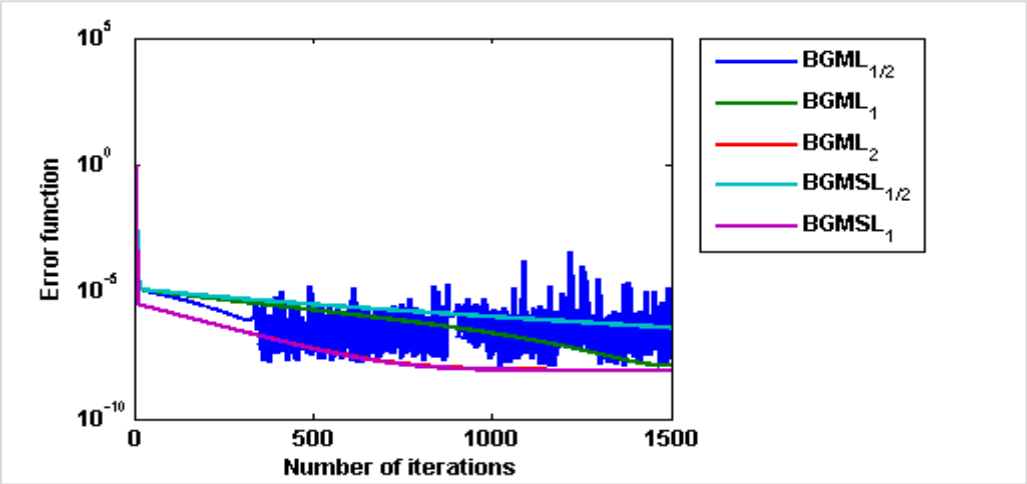

Figure 1 Comparison results of norm of gradient for Gaussian function approximation

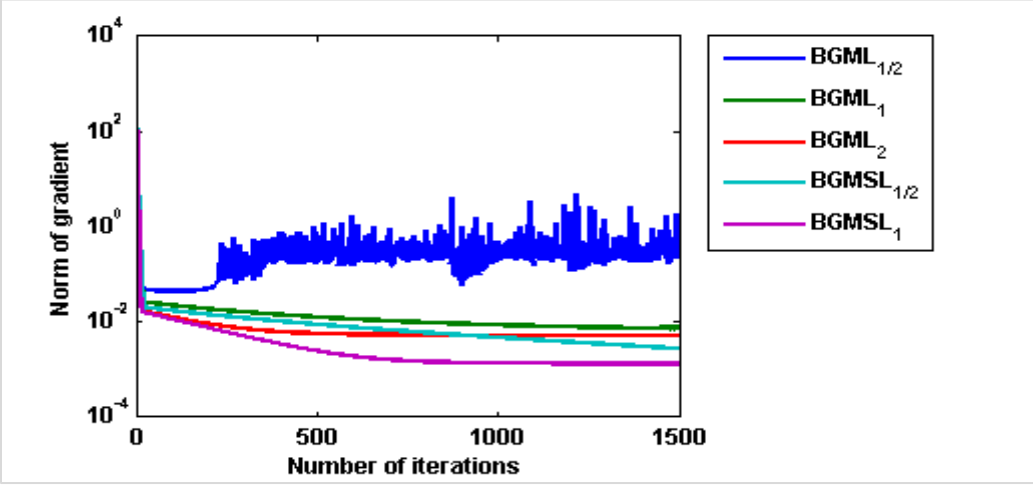

Figure 2 Comparison results of norm of gradient for Gaussian function approximation problem

The study introduces $OGMSL_1$ for FFNNs in order to presents some weak and strong convergence results for the learning methods, indicating that the gradient of the error function goes to zero and the weight sequence goes to a fixed point, respectively. To do so, $OGMSL_1$ compared with 5 different regularization methods: $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$. These models have been tested on monks and sonar classification datasets, higher-order Gaussian function problem, hyperbolic function problem, 4-bit and 5-bit parity problems. A complete list of abbreviations is listed in *Appendix II*.

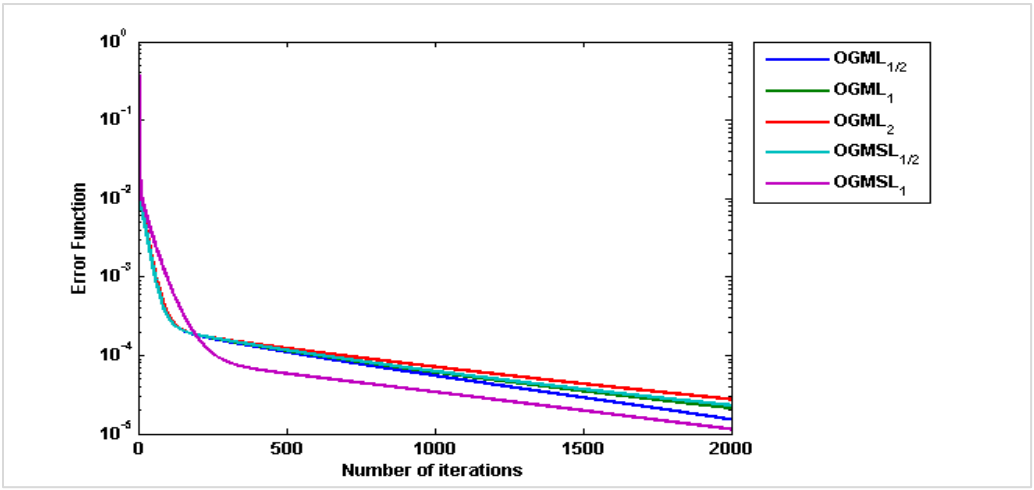

Figure 3 Comparison results of error function for hyperbolic function approximation problem

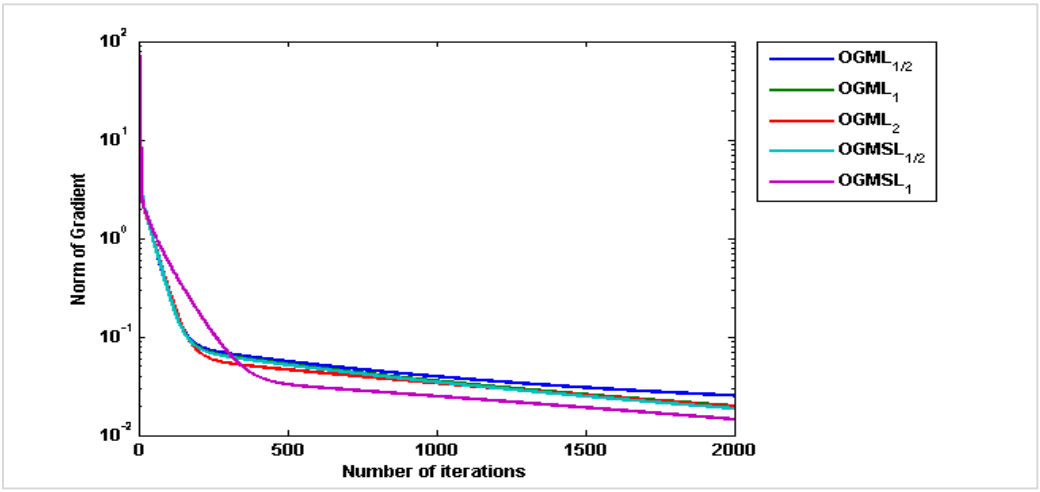

Figure 4 Comparison results of norm of gradient for hyperbolic function approximation problem

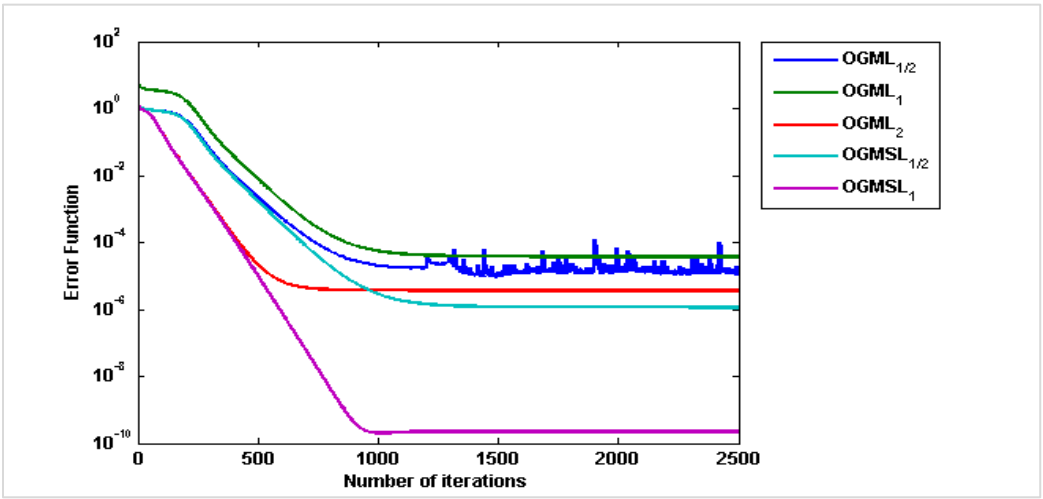

Figure 5 Results of the error function comparison for the 4-bit parity problem

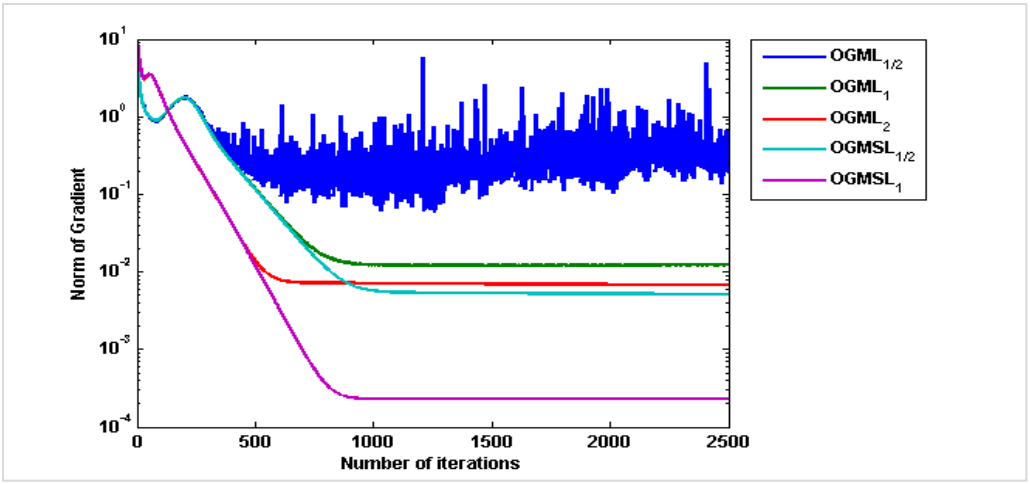

Figure 6 Results of the norm of gradient comparison for the 4-bit parity problem

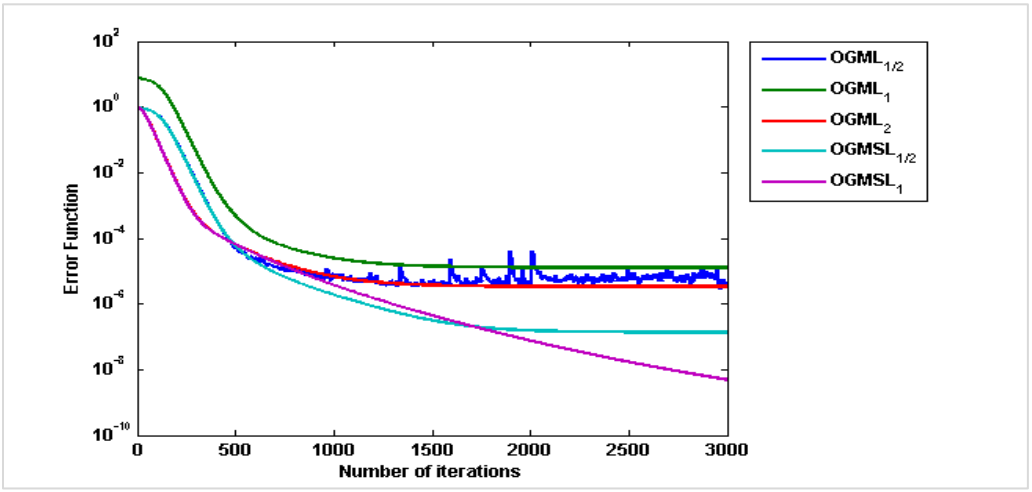

Figure 7 Results of the error function comparison for the 5-bit parity problem

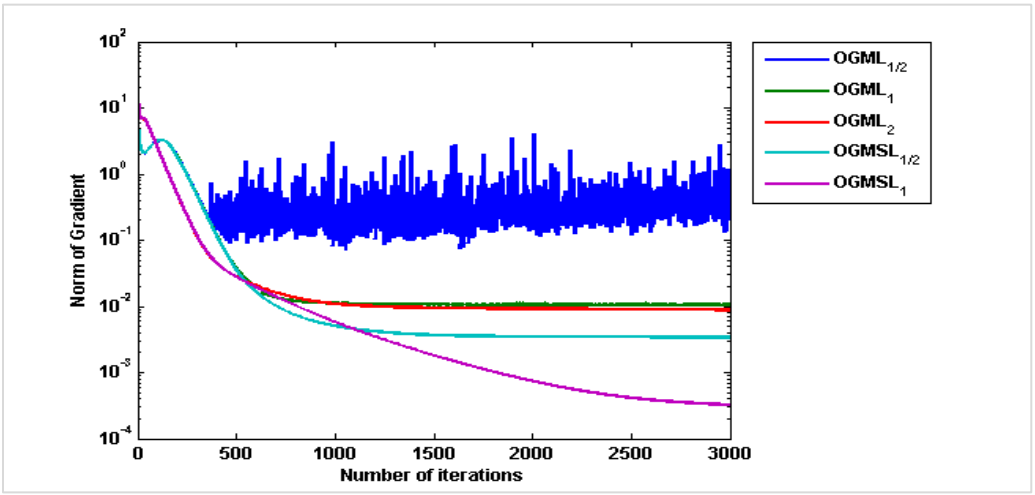

Figure 8 Results of the norm of gradient comparison for the 5-bit parity problem

Table 5 Comparison of performance when addressing the classification issues with Monk and Sonar problems

Problems	Algorithms	Training accuracy (%)	Testing accuracy (%)	Time (s)
Monk	$OGML_{1/2}$	88.73	79.11	4.282431
	$OGML_1$	89.23	79.41	4.277983
	$OGML_2$	92.36	87.78	4.251364
	$OGMSL_{1/2}$	95.09	90.25	4.270421
	$OGMSL_1$	97.98	94.11	4.093337
Sonar	$OGML_{1/2}$	86.12	82.28	122.871011
	$OGML_1$	85.61	81.16	124.090006
	$OGML_2$	83.45	80.71	123.039897
	$OGMSL_{1/2}$	88.37	83.14	122.616122
	$OGML_1$	90.76	89.45	122.544129

Table 6 Performance comparisons in dealing with Gaussian function approximation problem

Algorithms	Testing Error	Norm of Gradient	Time (%)
$OGML_{1/2}$	1.4458e-07	0.3837	6.651292
$OGML_1$	1.2776e-08	0.0077	6.609388
$OGML_2$	8.0474e-09	0.0052	6.586373
$OGMSL_{1/2}$	3.8483e-07	0.0027	6.661348
$OGMSL_1$	7.9309e-09	0.0013	6.527136

Table 7 Performance comparison in dealing with hyperbolic function approximation problem

Algorithms	Testing Error	Norm of Gradient	Time (%)
$OGML_{1/2}$	1.5209e-05	0.0253	12.732070
$OGML_1$	2.1175e-05	0.0198	12.791893
$OGML_2$	2.7493e-05	0.0194	12.920367
$OGMSL_{1/2}$	2.3672e-05	0.0458	12.830118
$OGMSL_1$	1.1418e-05	0.0145	12.731103

Table 8 Comparing the effectiveness of different approaches to 4- and 5-bit parity issues

Problems	Algorithms	Testing error	Norm of gradient	Time (%)
4-bit	$OGML_{1/2}$	1.2632e−05	0.2743	2.626743
	$OGML_1$	3.6483e−05	0.0124	2.629886
	$OGML_2$	3.6706e−06	0.0067	2.654304
	$OGMSL_{1/2}$	1.1512e−06	0.0051	2.673291
	$OGMSL_1$	2.2582e−10	0.0002	2.614614
5-bit	$OGML_{1/2}$	3.3780e−06	0.4609	8.072491
	$OGML_1$	1.2966e−05	0.0103	8.085458
	$OGML_2$	3.4186e−06	0.0089	8.121973
	$OGMSL_{1/2}$	1.3756e−07	0.0034	8.160070
	$OGML_1$	4.8959e−09	0.0003	8.034846

The findings are as under:

a. From *Table 5*, it is clear that the samples generated by $OGMSL_1$ are clearer and more realistic than other methods based on the monks and sonar datasets. Among $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$, $OGMSL_1$ produces better generalization performance results. Modified the usual L_1 regularization method ($OGML_1$) by smoothing it at the origin. This advantage helped to obtain the best results ever, which shows the extent of the influence of the smooth technique on controlling and reducing excessive weights, which $OGMSL_1$ achieved good accuracy for monk and sonar problems for training (97.98%; 94.11%) and testing (90.78%; 89.45%) respectively.

b. Though *Figures 1* to *8*, it is observed that the error functions of $OGMSL_1$ decrease monotonically,

and the error functions of $OGML_1$, $OGML_2$, $OGMSL_{1/2}$ are decrease, and $OGML_{1/2}$ also decrease but with oscillation. On the other hand, $OGMSL_{1/2}$ is the best, but the smoothing L_1 regularization method is undisputedly the best ever.

c. Furthermore, a relationship between the learning rate parameter and the penalty parameter and smoothing parameter is given to guarantee the convergence. Based on the *Tables 5* to *8*, it's obvious that $OGMSL_1$ gets processed more quickly because it requires less time.

Limitation of the study

The online gradient method is a popular optimization algorithm used to prune FFNNs. While it is widely used and effective, it has some limitations. The learning rate, which controls how quickly the algorithm updates the parameters, can significantly affect the convergence of the algorithm. If the learning rate is too high, the algorithm may oscillate or diverge; if it's too low, the algorithm may converge too slowly. The limitation of the proposed algorithm is that for particular parameters, the solution obtained by $OGMSL_1$ is better than the $OGML_{1/2}$, $OGML_1$, $OGML_2$, and $OGMSL_{1/2}$. If the parameters are varied then there is no surety for the best optimal solution. Thus, $OGMSL_1$ requires fine-tuning of parameters in order to get the best solutions for numerical experiments problems. To this end, learning was chosen to be constant.

5.Conclusion and future work

In the context of learning in FFNNs, the convergence of an $OGMSL_1$ has been examined in this study. A novel approach that has close resemblance to the existing convergence outcomes has been suggested. It is shown that the suggested algorithm converges to the set of zeroes of the gradient of the criterion error function with probability under relatively mild conditions. This method increases learning efficiency and streamlines the gradient of error function calculation as compared to previous findings. As a result, this strategy also has a more effective trimming impact. Furthermore, the convergence outcomes assumption is also a new expansion. Four numerical examples are presented to substantiate the claims. In our future study, the smooth regularization will be expanded and extended for use in other neural networks based on the results achieved.

Acknowledgment

The Researchers would like to thank the Deanship of Graduate Studies and Scientific Research at Qassim University for financial support (QU-APC-2024-9/1).

Conflicts of interest

The authors have no conflicts of interest to declare.

Data availability

The datasets used for the experimentation are publicly available:
MONK's: https://archive.ics.uci.edu/dataset/70
Sonar: https://archive.ics.uci.edu/dataset/151

Author's contribution statement

Khidir Shaib Mohamed: Conceptualization, writing original draft and editing, analysis and interpretation of results, review and supervision and **Suhail Abdullah Alsaqer:** Conceptualization, writing – review and editing.

References

[1] Montana DJ, Davis L. Training feedforward neural networks using genetic algorithms. In IJCAI 1989 (pp. 762-7).
[2] Jensen CA, Reed RD, Marks RJ, El-sharkawi MA, Jung JB, Miyamoto RT, et al. Inversion of feedforward neural networks: algorithms and applications. Proceedings of the IEEE. 1999; 87(9):1536-49.
[3] Nicole S. Feedforward neural networks for principal components extraction. Computational Statistics & Data Analysis. 2000; 33(4):425-37.
[4] Johansson EM, Dowla FU, Goodman DM. Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. International Journal of Neural Systems. 1991; 2(4):291-301.
[5] Cilimkovic M. Neural networks and back propagation algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin. 2015;15(1).
[6] Kushner HJ, Clark DS. Stochastic approximation methods for constrained and unconstrained systems. Springer Science & Business Media; 2012.
[7] Nevel'son MB, Has' minskii RZ. Stochastic approximation and recursive estimation. American Mathematical Society; 1976.
[8] Leinweber DJ. Stupid data miner tricks: overfitting the S&P 500. Journal of Investing. 2007; 16(1):15.
[9] Tetko IV, Livingstone DJ, Luik AI. Neural network studies. 1. Comparison of overfitting and overtraining. Journal of Chemical Information and Computer Sciences. 1995; 35(5):826-33.
[10] Caruana R, Lawrence S, Giles C. Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. Advances in Neural Information Processing Systems. 2000;13.
[11] Wan L, Zeiler M, Zhang S, Le CY, Fergus R. Regularization of neural networks using dropconnect. In international conference on machine learning 2013 (pp. 1058-66). PMLR.
[12] Matsuoka K. Noise injection into inputs in back-propagation learning. IEEE Transactions on Systems, Man, and Cybernetics. 1992; 22(3):436-40.

[13] Browne MW. Cross-validation methods. Journal of Mathematical Psychology. 2000; 44(1):108-32.
[14] Yu X, Chen Q. Convergence of gradient method with penalty for ridge polynomial neural network. Neurocomputing. 2012; 97:405-9.
[15] Zhou L, Fan Q, Huang X, Liu Y. Weak and strong convergence analysis of elman neural networks via weight decay regularization. Optimization. 2023; 72(9):2287-309.
[16] Shi G, Zhang J, Li H, Wang C. Enhance the performance of deep neural networks via L2 regularization on the input of activations. Neural Processing Letters. 2019; 50:57-75.
[17] Miao C, Yu H. Alternating Iteration for $ l_ {p} $($0< p\leq 1$) regularized CT reconstruction. IEEE Access. 2016; 4:4355-63.
[18] Zhang Z, Xu Y, Yang J, Li X, Zhang D. A survey of sparse representation: algorithms and applications. IEEE Access. 2015; 3:490-530.
[19] Ng AY. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In proceedings of the twenty-first international conference on machine learning 2004 (p. 78). ACM.
[20] Meng D, Zhao Q, Xu Z. Improve robustness of sparse PCA by L1-norm maximization. Pattern Recognition. 2012; 45(1):487-97.
[21] Xu Z, Chang X, Xu F, Zhang H. $ L_ {1/2} $ regularization: a thresholding representation theory and a fast solver. IEEE Transactions on Neural Networks and Learning Systems. 2012; 23(7):1013-27.
[22] He X, Sun Z. Sparse identification of dynamical systems by reweighted l1-regularized least absolute deviation regression. Communications in Nonlinear Science and Numerical Simulation. 2024; 131:107813.
[23] Gao J, Wang Y, Yao J, Zhan X, Sun G, Bai J. Three-dimensional array SAR sparse imaging based on hybrid regularization. IEEE Sensors Journal. 2024; 14(10): 16699- 709.
[24] Jiang Y, He Y, Zhang H. Variable selection with prior information for generalized linear models via the prior LASSO method. Journal of the American Statistical Association. 2016; 111(513):355-76.
[25] Tibshirani R. The lasso method for variable selection in the Cox model. Statistics in Medicine. 1997; 16(4):385-95.
[26] Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology. 1996; 58(1):267-88.
[27] Weigend AS, Rumelhart DE, Huberman BA. Back-propagation, weight-elimination and time series prediction. In connectionist models 1991(pp. 105-16). Morgan kaufmann.
[28] https://home.ttic.edu/~shai/papers/KakadeShalevTewari09.pdf. Accessed 11 April 2024.
[29] Zou H, Hastie T. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society Series B: Statistical Methodology. 2005; 67(2):301-20.
[30] Shaib K. Pruning feedforward polynomial neural with smoothing elastic net regularization. Sensors & Transducers. 2023; 260(1):14-23.
[31] Jiao J, Su K. A new sigma-Pi-sigma neural network based on L1 and L2 regularization and applications. AIMS Mathematics. 2024; 9(3):5995-6012.
[32] Zhang H, Tang Y. Online gradient method with smoothing ℓ0 regularization for feedforward neural networks. Neurocomputing. 2017; 224:1-8.
[33] Fan Q, Liu T. Smoothing l0 regularization for extreme learning machine. Mathematical Problems in Engineering. 2020; 2020(1):9175106.
[34] Mohamed KS, Mohammed YS. An online gradient method with smoothing L_0 regularization for Pi-sigma network. Transactions on Machine Learning and Artificial Intelligence. 2018; 6(6):96.
[35] Nguyen TT, Thang VD, Nguyen VT, Nguyen PT. SGD method for entropy error function with smoothing l 0 regularization for neural networks. Applied Intelligence. 2024:1-6.
[36] Fan Q, Zurada JM, Wu W. Convergence of online gradient method for feedforward neural networks with smoothing L1/2 regularization penalty. Neurocomputing. 2014; 131:208-16.
[37] Liu Y, Li Z, Yang D, Mohamed KS, Wang J, Wu W. Convergence of batch gradient learning algorithm with smoothing L1/2 regularization for sigma–Pi–sigma neural networks. Neurocomputing. 2015; 151:333-41.
[38] Zhao J, Zurada JM, Yang J, Wu W. The convergence analysis of spikeprop algorithm with smoothing L1⁄ 2 regularization. Neural Networks. 2018; 103:19-28.
[39] Lu Y, Li W, Wang H. A batch variable learning rate gradient descent algorithm with the smoothing L 1/2 regularization for takagi-sugeno models. IEEE Access. 2020; 8:100185-93.
[40] Xie X, Zhang H, Wang J, Chang Q, Wang J, Pal NR. Learning optimized structure of neural networks by hidden node pruning with $ L_ {1} $ regularization. IEEE Transactions on Cybernetics. 2019; 50(3):1333-46.
[41] Onaran I, Ince NF, Cetin AE. Sparse spatial filter via a novel objective function minimization with smooth ℓ1 regularization. Biomedical Signal Processing and Control. 2013; 8(3):282-8.
[42] Schmidt M, Fung G, Rosales R. Fast optimization methods for l1 regularization: a comparative study and two new approaches. In machine learning: ECML: 18th European conference on machine learning, Warsaw, Poland 2007 (pp. 286-97). Springer Berlin Heidelberg.
[43] Mohamed KS. Batch gradient learning algorithm with smoothing L1 regularization for feedforward neural networks. Computers. 2022; 12(1):1-15.
[44] Tadic V, Stankovic S. Learning in neural networks by normalized stochastic gradient algorithm: local convergence. In proceedings of the 5th seminar on neural network applications in electrical engineering, NEUREL 2000 (pp. 11-7). IEEE.
[45] Xu ZB, Zhang R, Jing WF. When does online BP training converge? IEEE Transactions on Neural Networks. 2009; 20(10):1529-39.

[46] Wu W, Feng G, Li Z, Xu Y. Deterministic convergence of an online gradient method for BP neural networks. IEEE Transactions on Neural Networks. 2005; 16(3):533-40.
[47] Wu W, Xu Y. Deterministic convergence of an online gradient method for neural networks. Journal of Computational and Applied Mathematics. 2002; 144(1-2):335-47.
[48] Li Z, Wu W, Tian Y. Convergence of an online gradient method for feedforward neural networks with stochastic inputs. Journal of Computational and Applied Mathematics. 2004; 163(1):165-76.
[49] Wu W, Feng G, Li X. Training multilayer perceptrons via minimization of sum of ridge functions. Advances in Computational Mathematics. 2002; 17:331-47.
[50] Wu W, Wang J, Cheng M, Li Z. Convergence analysis of online gradient method for BP neural networks. Neural Networks. 2011; 24(1):91-8.
[51] Wang J, Wu W, Zurada JM. Deterministic convergence of conjugate gradient method for feedforward neural networks. Neurocomputing. 2011; 74(14-15):2368-76.

Khidir Shaib Mohamed received his M.S. degree in Applied Mathematics from Jilin University, Changchun, China, in 2011, and his Ph.D. degree in Computational Mathematics from Dalian University of Technology, Dalian, China, in 2018. His research interests include Theoretical Analysis and Regularization Methods for Neural Networks and Machine Learning.
Email: K.Idris@qu.edu.sa

Suhail Abdullah Alsaqer received his M.S. degree in Applied Computer Science from Gannon University, Erie, Pennsylvania, USA, in 2014, and another M.S. degree in Engineering Management from the same institution. His research interests include Artificial Intelligence, Image Processing, Machine Learning, and Computer Vision.
Email: s.alsaqer@qu.edu.sa

Appendix I

To make things easier, we present the following notations from Equations 23 to 28.

$$\boldsymbol{G^{nJ+j,j}} = \boldsymbol{G}(\boldsymbol{\mathcal{V}^{nJ+j}\xi^{j}}) \quad (23)$$

$$\psi^{n,j,j} = G^{nJ+j,j} - G^{nJ,j} \quad (24)$$

$$r_i^{n,j} = \Delta_j^n w_i^{nJ+j-1} - \Delta_j^n w_i^{nJ} \quad (25)$$

$$d_i^{n,j} = w_i^{nJ+j} - w_i^{nJ} = \sum_{j=1}^{J} \Delta_j^n w_i^{nJ+j-1} = \sum_{j=1}^{J} \left(\Delta_j^n w_i^{nJ} + r_i^{n,j}\right) \quad (26)$$

for $n \in \mathbb{N}$, $i = 1,2,\cdots$; $j = 1,2,\cdots,J$.
Let constants C_1 and C_2 be defined by (cf. Assumption 3), we get

$$C_1 = \max_{1\le j\le J}\left\{\left\|\xi^j\right\|, \left|O^j\right|\right\}, \qquad C_2 = \sup_{n\in\mathbb{N}}\left\|\mathcal{W}^{nJ+j}\right\| \quad (27)$$

For $j = 1,2,\cdots,J$, $g_j'(t)$ likewise meets the Lipschitz condition according to assumption 1. Furthermore, on any limited closed interval, $g(t)$ and $g_j(t)$ are uniformly continuous. We've got

$$C_3 = max\begin{Bmatrix} \sup_{t\in\mathbb{R}}|g(t)|, \sup_{t\in\mathbb{R}}|g'(t)|, \sup_{t\in\mathbb{R},1\le j\le J}\left|g_j'(t)\right| \\ , \sup_{t\in\mathbb{R}}|g''(t)|, \sup_{t\in\mathbb{R},1\le j\le J}\left|g_j''(t)\right| \end{Bmatrix} \quad (28)$$

Lemma 1. Assume that Assumptions 1 and 2 are true, and that Equation 13, which generates the sequence $\{\mathcal{W}^{\mathrm{nJ+j}}\}$, is genuine. Next, constants $C_4 - C_7$ exist such that from Equations 27 to 30.

$$\left\|G^{n+j,j}\right\| \le C_4, \quad (29)$$

$$\left\|\psi^{n,j,j}\right\| \le C_5\eta_n, \quad (30)$$

$$\left\|d_i^{n,j}\right\| \le C_6\eta_n, \quad (31)$$

$$\left\|r_i^{n,j}\right\| \le C_7\eta_n, \quad (32)$$

where $\in \mathbb{N}$ $i = 0,1,2,\cdots; j = 1,2,\cdots,J$.

Proof. The proof is not included here because it is essentially the same as the one found in Lemma 4.4 of [50].

Lemma 2. Allow Equation 16 to produce the sequence $\{\mathcal{W}^{nJ+j}\}$. Let us assume Assumptions 1 through 3 as shown in Equation 31.

$$E\left(\mathcal{W}^{(n+1)J}\right) \le E(\mathcal{W}^{nJ}) - \|E_{\mathcal{W}}(\mathcal{W}^{nJ})\|^2 + C_8\eta_n^2 \quad (33)$$

where $C_8 > 0$ is a constant that is unaffected by n or η_n.
Proof. Assumption 2, Equation 28 and using the Taylor formula to expand $f\left(w_i^{(n+1)J}\right)$ give one that in Equation 33.

$$\begin{aligned} &\boldsymbol{f}\left(\boldsymbol{w}_i^{(n+1)J}\right) \le \boldsymbol{f}\left(\boldsymbol{w}_i^{nJ}\right) + \left(\boldsymbol{f}'\left(\boldsymbol{w}_i^{nJ}\right)\right)^T \\ &\times \sum_{i=1}^{q}\left(\boldsymbol{w}_i^{(n+1)J} - \boldsymbol{w}_i^{nJ}\right) \\ &+ \frac{1}{2}\boldsymbol{f}''\left(\boldsymbol{t}_i^{n,j}\right)\sum_{i=1}^{q}\left\|\boldsymbol{w}_i^{(n+1)J} - \boldsymbol{w}_i^{nJ}\right\|^2 \\ &= \boldsymbol{f}\left(\boldsymbol{w}_i^{nJ}\right) + \sum_{i=1}^{q}\boldsymbol{f}'\left(\boldsymbol{w}_i^{nJ}\right)\cdot\ \boldsymbol{d}_i^{n,J} \\ &+ \tfrac{1}{2}\boldsymbol{f}''\left(\boldsymbol{t}_i^{nj}\right)\textstyle\sum_{i=1}^{q}\left\|\ \boldsymbol{d}_i^{nj}\right\|^2 \end{aligned} \quad (34)$$

By virtue of Assumption 1 and Lemma 1, we know that the derivative $\boldsymbol{g}''\left(\boldsymbol{w}_i^{nJ}\cdot\boldsymbol{\xi}^j + \boldsymbol{t}(\ \boldsymbol{d}_i^{n,j}\cdot\boldsymbol{\xi}^j)\right)$ is integrable almost everywhere on $[\mathbf{0},\mathbf{1}]$ and in Equation 34.

$$\begin{aligned} &\boldsymbol{w}_0^{nJ}\cdot\ \boldsymbol{\psi}^{n,J,j} \\ &= \sum_{i=1}^{q}\boldsymbol{w}_{0i}^{nJ}\left[\boldsymbol{g}\left(\boldsymbol{w}_i^{(n+1)J}\cdot\boldsymbol{\xi}^j\right) - \boldsymbol{g}\left(\boldsymbol{w}_i^{nJ}\cdot\boldsymbol{\xi}^j\right)\right] \\ &= \sum_{i=1}^{q}\boldsymbol{w}_{0i}^{nJ}\boldsymbol{g}\left(\boldsymbol{w}_i^{nJ}\cdot\boldsymbol{\xi}^j\right)\boldsymbol{d}_i^{n,j}\cdot\boldsymbol{\xi}^j \\ &+ \sum_{i=1}^{q}\boldsymbol{w}_{0i}^{nJ}\left(\ \boldsymbol{d}_i^{n,j}\cdot\boldsymbol{\xi}^j\right)^2\int_0^1(\mathbf{1}-\boldsymbol{t}) \\ &\times \boldsymbol{g}''\left(\boldsymbol{w}_i^{nJ}\cdot\boldsymbol{\xi}^j + \boldsymbol{t}(\ \boldsymbol{d}_i^{n,j}\cdot\boldsymbol{\xi}^j)\right)\boldsymbol{dt} \end{aligned} \quad (35)$$

By Lemma 1, Equations 18, 19 and 35, a constant $C_{8,1} > 0$ exists such that in Equation 34.

$$\begin{aligned} &\boldsymbol{g}_j\left(\boldsymbol{w}_0^{(n+1)J}\cdot\boldsymbol{G}^{(n+1)Jj}\right) \\ &\le \boldsymbol{g}_j\left(\boldsymbol{w}_0^{nJ}\cdot\boldsymbol{G}^{nJ,j}\right) + \boldsymbol{g}_j'\left(\boldsymbol{w}_0^{nJ}\cdot\boldsymbol{G}^{nJ,j}\right) \\ &\times\left[\boldsymbol{w}_0^{(n+1)J}\cdot\boldsymbol{G}^{(n+1)Jj} - \boldsymbol{w}_0^{nJ}\cdot\boldsymbol{G}^{nJ,j}\right] \\ &+\boldsymbol{C}_{8,1}\left(\boldsymbol{w}_0^{(n+1)J}\cdot\boldsymbol{G}^{(n+1)Jj} - \boldsymbol{w}_0^{nJ}\cdot\boldsymbol{G}^{nJ,j}\right)^2 \end{aligned}$$

$$
\begin{aligned}
&= g_j\left(w_0^{nJ}\cdot G^{nJ,j}\right) + g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\\
&\times\left[d_0^{n,j}\cdot G^{nJ,\tau} + w_0^{nJ}\cdot \psi^{nJ,j} + d_0^{n,j}\cdot \psi^{nJ,j}\right]\\
&+C_{8,1}\left(d_0^{nJ}\cdot G^{nJ,j} + w_0^{nJ}\cdot \psi^{nJ,j} + d_0^{n,j}\cdot \psi^{nJ,j}\right)^2\\
&= g_j\left(w_0^{nJ}\cdot G^{nJ,j}\right) + g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)G^{nJ,j}\cdot d_0^{n,j}\\
&+g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\sum_{i=1}^{q} w_{0i}^{nJ}\, g'\left(w_i^{nJ}\cdot \xi^j\right)\xi^j\cdot d_i^{n,j}\\
&+g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\sum_{i=1}^{q} w_{0i}^{nJ}\left(d_i^{n,j}\cdot \xi^j\right)^2\\
&\times\int_0^1 (1-t)g''\left(w_i^{nJ}\cdot \xi^j + t\left(d_i^{n,j}\cdot \xi^j\right)\right)dt\\
&+g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\psi^{n,j,j}\cdot d_0^{n,j}\\
&+C_{8,1}\left[d_0^{n,j}\cdot G^{nJ,j} + w_0^{nJ}\cdot \psi^{n,J,j} + d_0^{n,j}\cdot \psi^{nJ,j}\right]^2
\end{aligned}
\tag{36}
$$

Then, by the error function in Equation 13. We write in Equation 37.

$$
\begin{aligned}
&E\left(\mathcal{W}^{(n+1)J}\right) = \sum_{j,\tau=1}^{J} g_j\left(w_0^{(n+1)J}\cdot G^{(n+1)J,j}\right)\\
&+\lambda\textstyle\sum_{i=1}^{q}\sum_{k=0}^{p} f\left(w_{ik}^{(n+1)J}\right)
\end{aligned}
\tag{37}
$$

Combining E Equations 35, 36 and using Equation 37, and for $k = 0,1,2,\cdots,p$. We have

$$
\begin{aligned}
&E\left(\mathcal{W}^{(n+1)J}\right)\\
&= \sum_{j=1}^{J} g_j\left(w_0^{(n+1)J}\cdot G^{(n+1)J,j}\right) + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p} f\left(w_{ik}^{(n+1)J}\right)\\
&\le \sum_{j=1}^{J} g_j\left(w_0^{nJ}\cdot G^{nJ}\right) + d_0^{n,j}\cdot\left(\sum_{j=1}^{J} g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)G^{nJ,j}\right)\\
&+\sum_{i=1}^{p} d_i^{n,j}\left[\sum_{j=1}^{J} g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)w_{0i}^{nJ}\; g'\left(w_i^{nJ}\cdot \xi^j\right)\right] + \delta_1\\
&+\lambda\sum_{k=0}^{p}\sum_{j=1}^{q}\left[f\left(w_{ik}^{nJ}\right) + f'\left(w_{ik}^{nJ}\right)\cdot d_{ik}^{n,j}\right]\\
&+\frac{1}{2}\lambda\sum_{k=0}^{p}\sum_{i=1}^{q}\left[f''\left(t_i^{n,j}\right)\left(d_{ik}^{n,j}\right)^2\right]\\
&= \sum_{j=1}^{J} g_j\left(w_0^{nJ}\cdot G^{nJ}\right) + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p} f\left(w_{ik}^{nJ}\right)\\
&+\sum_{i=1}^{q}\left(\sum_{j,\tau=1}^{J} g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)G^{nJ,j}\right)d_{i0}^{nJ}\\
&+\lambda\sum_{i=1}^{q}\left(\sum_{j=1}^{J} f'\left(w_{i0}^{nJ}\right)\right)\cdot d_{i0}^{n,j}\\
&+\sum_{i=1}^{q}\sum_{k=1}^{p} d_{ik}^{n,j}\cdot\left(\sum_{j=1}^{J} g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\right)\\
&\times w_{0i}^{nJ} g'^{\left(w_i^{nJ}\cdot\xi^j\right)}\xi_k^j\\
&+\lambda\sum_{i=1}^{q}\sum_{k=1}^{p} d_{ik}^{n,j}\cdot\left(\sum_{j=1}^{J} f'^{\left(w_{ik}^{nJ}\right)}\right)\\
&+\frac{1}{2}\lambda f''\left(t_i^{n,j}\right)\sum_{k=0}^{p}\sum_{i=1}^{q}\left\| d_{ik}^{n,j}\right\|^2 + \delta_1\\
&= E\left(\mathcal{W}^{nJ}\right) + \sum_{k=0}^{p}\sum_{i=1}^{q}\left(\sum_{j=1}^{J}\Delta_j^n w_{ik}^{nJ}\right)\cdot E_{w_{ik}}\left(\mathcal{W}^{nJ}\right)\\
&+\sum_{k=0}^{p}\sum_{i=1}^{q}\left(\sum_{j=1}^{J} r_{ik}^{n,j}\right)\cdot E_{w_{ik}}\left(\mathcal{W}^{nJ}\right)\\
&+\frac{1}{2}\lambda f''\left(t_i^{nj}\right)\sum_{k=0}^{p}\sum_{\substack{j=1\\ p}}^{q}\left\| d_{ik}^{nj}\right\|^2 + \delta_1\\
&= E\left(\mathcal{W}^{nJ}\right) - \eta_n\sum_{k=0}^{p}\sum_{i=1}^{q}\left(E_{w_{ik}}\left(\mathcal{W}^{nJ}\right)\cdot E_{w_{ik}}\left(\mathcal{W}^{nJ}\right)\right)\\
&+\delta_1 + \delta_2\\
&= E\left(\mathcal{W}^{nJ}\right) - \eta_n\left\|E_{\mathcal{W}}\left(\mathcal{W}^{nJ}\right)\right\|^2 + \delta_1 + \delta_2
\end{aligned}
\tag{38}
$$

where

$$
\begin{aligned}
&\delta_1 = g_j\left(w_0^{nJ}\cdot G^{nJ,j}\right) + g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)G^{nJ,j}\cdot d_0^{nJ}\\
&+g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\sum_{i=1}^{q}\sum_{k=1}^{p} w_{0i}^{nJ}\, g'\left(w_i^{nJ}\cdot \xi^j\right)\xi^j\cdot d_{ik}^{nJ}\\
&+g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\sum_{i=1}^{q}\sum_{k=0}^{p} w_{0i}^{nJ}\left(d_{ik}^{n,j}\cdot \xi^j\right)^2\\
&\times\int_0^1 (1-t)g''\left(w_i^{nJ}\cdot \xi^j + t\left(d_{ik}^{n,j}\cdot \xi^j\right)\right)dt\\
&+g_j'\left(w_0^{nJ}\cdot G^{nJ,j}\right)\psi^{n,j,j}\cdot d_0^{n,j}\\
&+C_{8,1}\left(d_0^{n,j}\cdot G^{nJ,j} + w_0^{nJ}\cdot \psi^{n,j,j} + d_0^{n,j}\cdot \psi^{nJ,j}\right)^2
\end{aligned}
\tag{39}
$$

and

$$
\delta_2 = C_{8,2}\lambda\textstyle\sum_{k=1}^{p}\sum_{i=1}^{q}\left\| d_{ik}^{nj}\right\|^2 + \sum_{k=1}^{p}\sum_{i=1}^{q}\sum_{j=1}^{J} r_{ik}^{n,j}\cdot E_{w_{ik}}\left(\mathcal{W}^{nJ}\right)
\tag{40}
$$

where $C_{8,2} = \frac{1}{2}f''\left(t_i^{n,j}\right)$ and in between $w_i^{(n+1)J}$ and w_i^{nJ} lies a constant called $t_i^{n,j} \in \mathbb{R}$.

It is important to remember the underlying Assumptions 1 through 3 which ensure the boundedness of the network weights, the error function, the gradient, the active function g and its first and second derivatives, and the active function. Therefore, it is simple to demonstrate that a constant exists by considering Equation 27 to Equation 30 and using the triangle inequality and Cauchy-Schwartz inequality.

$$
\delta_n = C_8\eta_n^2
\tag{41}
$$

where $\delta_n = \delta_1 + \delta_2$.Thus, combining Equations 36 and 39 yields the appropriate estimate.

Lemma 3. (see Lemma 1 in [45]) Let $\{\mathcal{M}_m\}$, $\{\mathcal{W}_m\}$ and $\{\mathcal{N}_m\}$ be three sequences such that $\mathcal{W}_m$ is nonnegative for all $m(m = 0,1,2,\cdots)$. Assume that

$$
\mathcal{M}_{m+1} \le \mathcal{M}_m - \mathcal{W}_m + \mathcal{N}_m
\tag{42}
$$

as well as the convergence of the series $\sum_{m=0}^{\infty}\mathcal{N}_m$. After that, $\sum_{m=0}^{\infty}\mathcal{W}_m < \infty$ and $\mathcal{M}_m \to -\infty$, respectively, as $\mathcal{M}_m$ converges to a finite value.

Lemma 4. (see Lemma 4.2 in [50]) Suppose that the learning rate η_n satisfies Assumption (A4) and that the sequence $a_n(n \in \mathbb{N})$satisfies $a_n \ge 0$, $\sum_{n=0}^{\infty}\eta_n a_n^{\beta} < \infty$ and $|a_{n+1} - a_n| \le \mu\eta_n$ for some positive constant β and μ that are positive. Next, we obtain $\lim_{n\to\infty} a_n = 0$.

Lemma 5. (see Lemma 5.3 in [51]) For a bounded closed region $\mathcal{A}$, let $F\colon \mathcal{A} \subset \mathbb{R}^m$ be continuous, and let $\mathcal{B} = \{\mathcal{Z} \in \mathcal{A}\colon F(\mathcal{Z}) = 0\}$. There are no interior points in the projection of $\mathcal{B}$ on any of the coordinate axes. Permit the sequence $\{\mathcal{Z}\}$ to fulfill:

$$
\lim_{n\to\infty} F\left(\mathcal{Z}^n\right) = 0, \qquad \lim_{n\to\infty}\left|\mathcal{Z}^{n-1} - \mathcal{Z}^n\right| = 0
\tag{43}
$$

After that, $\lim_{n\to\infty}\mathcal{Z}^n = \mathcal{Z}^*$ for any unique $\mathcal{Z}^* \in \mathcal{B}$.

It is now possible to demonstrate Theorem 1.

Proof of Theorem 1. By the Assumption 2, Lemma 2 and Lemma 3, we can see that

$$\sum_{n=0}^{\infty}\eta_n\|E_{\mathcal{W}}(\mathcal{W}^{nJ})\|^2=\eta_n\sum_{k=0}^{p}\sum_{i=1}^{q}\left\|E_{w_{ik}}(\mathcal{W}^{nJ})\right\|^2<\infty \quad (44)$$

Using the series' comparison discriminant technique. We've got

$$\sum_{n=0}^{\infty}\eta_n\left\|E_{w_{ik}}(\mathcal{W}^{nJ})\right\|^2<\infty \quad (45)$$

It is similar estimation Lemma 3, there exists a suitable constant C_9 such that

$$\left\|\boldsymbol{E_{w_{ik}}}\left(\boldsymbol{\mathcal{W}^{(n+1)J}}\right)-\boldsymbol{E_{w_{ik}}}(\boldsymbol{\mathcal{W}^{nJ}})\right\|=\boldsymbol{C_{9,1}}\left\|\boldsymbol{w_{ik}^{(n+1)J}}-\boldsymbol{w_{ik}^{nJ}}\right\|$$

$$=\boldsymbol{C_{9,1}}\sum_{i=1}^{q}\sum_{k=0}^{p}\left\|\boldsymbol{d_{ik}^{n,j+1}}\right\|$$

$$\leq JC_6C_{9,1}\,\eta_n=C_9\eta_n \quad (46)$$

where $\boldsymbol{C_9=Jmax\{C_6,C_{9,1}\}}$, $\boldsymbol{n=0,1,2,\cdots}$, and $\boldsymbol{j=1,2,\cdots,J}$.

Using Lemma 4 and the two aforementioned Equation 45 and Equation 46, you can find

$$\lim_{n\to\infty}E_{w_{ik}}(\mathcal{W}^{nJ})=0 \quad (47)$$

Considering the Assumption 2, it is easy to have

$$\boldsymbol{\lim_{n\to\infty}\eta_n=0} \quad (48)$$

By the Equation 46, we obtain

$$\left\|E_{w_{ik}}(\mathcal{W}^{(n+1)J})\right\|\leq\left\|E_{w_{ik}}(\mathcal{W}^{(n+1)J})-E_{w_{ik}}(\mathcal{W}^{nJ})\right.$$
$$\left.+E_{w_{ik}}(\mathcal{W}^{nJ})\right\|\leq C_9\eta_n+\left\|E_{w_{ik}}(\mathcal{W}^{(n+1)J})\right\| \quad (49)$$

Thus, for $\boldsymbol{n=0,1,2,\cdots,n=0.1.2,\cdots,q,k=1,2,\cdots,p,}$ and $\boldsymbol{j=1,2,\cdots,J}$

$$\boldsymbol{\lim_{n\to\infty}E_{w_{ik}}(\mathcal{W}^{nJ+j})=0} \quad (50)$$

Hence,

$$\lim_{n\to\infty}\left\|E_{w_{ik}}(\mathcal{W}^{nJ+j})\right\|=0 \quad (51)$$

Thus, the weak convergence of Theorem 1 is proved using Equation 19.

Now we begin to prove the strong convergence Equation 19 of Theorem 1.

Keep in mind that the error function $\mathrm{E}(\mathcal{W})$, as described in Equation 14 is both differentiable and continuous. Using Equations 51 and 31 with Equations 23 through 26, we obtain

$$\|\boldsymbol{\mathcal{W}^{nJ+j}-\mathcal{W}^{nJ+j-1}}\|^2=\sum_{i=1}^{q}\sum_{k=0}^{p}\left\|\boldsymbol{d_{ik}^{n,j}-d_{ik}^{n,j+1}}\right\|^2\leq$$

$$\sum_{i=1}^{q}\sum_{k=0}^{p}\boldsymbol{2}\left(\left\|\boldsymbol{d_{ik}^{n,j}}\right\|^2+\left\|\boldsymbol{d_{ik}^{n,j+1}}\right\|^2\right)$$

i.e.

$$\boldsymbol{\lim_{n\to\infty}\|\mathcal{W}^{nJ+j}-\mathcal{W}^{nJ+j-1}\|=0}. \quad (52)$$

According to the Assumption 4, Equations 51, 52 and Lemma 5, there exists a point $\mathcal{W}^*\in\Theta_0$, such that

$$\lim_{n\to\infty}\mathcal{W}^{nJ+j}=\mathcal{W}^*. \quad (53)$$

This proof is completed.

Appendix II

S. No.	Abbreviation	Description
1	BP	Backpropagation
2	FFNNs	Feedforward neural networks
3	LASSO	Least Absolute Shrinkage and Selection Operator
4	$OGMSL_1$	Online Gradient Method with Smoothing L_1 Regularization
5	$OGML_1$	Online Gradient Method With L_1 Regularization
6	$OGML_2$	Online Gradient Method with Smoothing L_2 Regularization
7	$OGML_{1/2}$	Online Gradient Method With $L_{1/2}$ Regularization
8	UCI	University of California, Irvine