# Convergence Analyses on Sparse Feedforward Neural Networks via Group Lasso Regularization ☆

Jian Wang[a,d,*], Qingling Cai[b], Qingquan Chang[c], Jacek M. Zurada[d,e]

[a] *College of Science, China University of Petroleum, Qingdao, 266580, China*
[b] *School of Engineering, Sun Yat-sen University, Guangzhou, 510275, China*
[c] *School of Mathematics and Statistics, Lanzhou University, Lanzhou, 730000, China*
[d] *Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY, 40292, USA*
[e] *Information Technology Institute, University of Social Sciences, Łódź 90-113, Poland*

## Abstract

In this paper, a new variant of feedforward neural networks has been proposed for a class of nonsmooth optimization problems. The penalty term of the presented neural networks stems from the Group Lasso method which selects hidden variables in a grouped manner. To deal with the non-differentiability of the original penalty term ($l_1$-$l_2$ norm) and avoid oscillations, smoothing techniques have been used to approximate the objective function. It is assumed that the training samples are supplied to the networks in a specific incremental way during training, that is, in each cycle samples are supplied in a fixed order. Then, under suitable assumptions on learning rate, penalization coefficients and smoothing parameters, the weak and strong convergence of the training process for the smoothing neural networks have been proved. The convergence analysis shows that the gradient of the smoothing error function approaches zero and the weight sequence converges to a fixed point, respectively. We demonstrate how the smoothing approximation parameter

☆This workwas supported in part by the National Natural Science Foundation of China (No. 61305075), the China Postdoctoral Science Foundation (No. 2012M520624), the Natural Science Foundation of Shandong Province (No. ZR2013FQ004), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20130133120014) and the Fundamental Research Funds for the Central Universities (No. 13CX05016A, 14CX05042A, 15CX02079A, 15CX05053A, 15CX08011A).

*Corresponding author
*Email addresses:* wangjiannl@upc.edu.cn (Jian Wang), jacek.zurada@louisville.edu (Jacek M. Zurada)

can be updated in the training procedure so as to guarantee the convergence of the procedure to a Clarke stationary point of the original optimization problem. In addition, we have proved that the original nonsmoothing algorithm with $l_1 - l_2$ norm penalty converges consistently to the same optimum solution with the corresponding smoothed algorithm. Numerical simulations demonstrate the convergence and effectiveness of the proposed training algorithm.

*Keywords:* Clarke gradient; Convergence; Feedforward neural networks; Group Lasso; Non-differentiability.

## 1. Introduction

Artificial neural networks have been widely used in various applications, such as pattern recognition, machine learning, data mining and signal processing [37, 27, 24]. The feedforward networks are one of the most popular architectures for their strong structural flexibility, good representation ability and compatibility with different training algorithms.

A reasonable architecture is one of the key aspects to guarantee better generalization of the trained neural networks [2]. Generally, the number of neurons in the input and output layers is fixed and represents the attributes and the target values of the dataset, respectively. Whereas the number of neurons in the hidden layer (or layers) depends on the complexity of the problem to be modeled, it is essential how to optimize the number of hidden neurons.

Typically, there are two approaches to determine the number of hidden neurons. One is the growing method, which starts with a small initial network and then adds hidden neurons stepwise during training [58, 35, 39].

The other is a pruning way, which starts with a large initial network and then removes the unnecessary neurons or weights [29, 40, 38, 56, 45, 1]. Researchers have developed a number of pruning algorithms to optimize network architectures. An interesting comparative study for pruning algorithms of neural networks has been presented in [2]. Based on the elimination techniques, pruning methods could be categorized as penalty term methods, cross validation methods [30], magnitude base methods [21], evolutionary pruning methods [47], mutual information (MI) [18], significance based pruning methods and sensitivity analysis (SA) method [3, 1]. This paper focuses on pruning technique by using a novel penalty term for backpropagation (BP) neural networks.

For standard networks, the error function is defined as the sum of the squared errors

$$E = \frac{1}{2}\sum_{j} \|Output^{j} - Target^{j}\|^{2}, \tag{1}$$

where $j$ represents the $j$-th sample of the dataset. When a network is to be pruned, it is common to add a penalty term to the error function during training

$$E = \frac{1}{2}\sum_{j} \|Output^{j} - Target^{j}\|^{2} + \lambda\phi(\mathbf{w}). \tag{2}$$

The penalty term is to suppress the unnecessary connections between neurons. The parameter $\lambda > 0$ is the regularization coefficient, which balances the relative importance of the penalty term and the pure error expression.

There are three typical regularization forms for feedforward networks: *weight decay* [9, 26, 41, 45], *weight elimination* [40, 46, 34, 31] and *approximate smoother* [36, 15].

When the backpropagation method is employed to train the network, uniform weight decay in [26] has a disadvantage that large weights are decaying at the same rate as small weights [22]. To remedy this problem, the weight elimination method has been suggested in [46]. Unfortunately, it does not distinguish between the large and very large weights [40]. The approximate smoother [50] appears to be more accurate than weight decay or weight elimination from the complexity point of view. It is designed for a multilayer perceptron with a single hidden layer and a single output neuron. However, it is more complicated than weight decay or weight elimination and has additional computational cost [23].

We notice that the above penalty methods discourage specific connections among neurons. To improve the interpretability and sparsity of neural networks, we will borrow the idea of Group Lasso to optimize the network architecture in this paper.

Lasso, the "least absolute shrinkage and selection operator" was first proposed for linear regression problem as a new technique that reduces some coefficients and sets others to zero [44]. It has been popular for simultaneous estimation and variable selection. However, lasso often results in selecting more factors than necessary and the solution depends on how the factors are represented. Then, a new version of the lasso, the adaptive lasso, was proposed in [59] by employing adaptive weights with $l_1$ penalty term. In addition, it enjoys the oracle properties.

An extension of lasso known as Group Lasso has been developed in [57] which selects the final models on the solution paths and encourages sparsity at group level. The penalty function ($l_1$-$l_2$ norm) is intermediate between the $l_1$ penalty in lasso and the $l_2$ penalty in ridge regression (weight decay). In addition, a more general form, sparse Group Lasso, has been investigated in [19] which blends the lasso with the Group Lasso. Its main advantage is that it yields the sparse solutions at both the group and individual feature levels.

A novel idea of this paper is to replace the common penalty terms in [26] with a Group Lasso counterpart for BP networks. We expect that this can enhance some of the desirable properties of Group Lasso and improve the pruning performance with better generalization.

For a multi-layer network, we denote by $\mathbf{W}_i$ the weight matrix connecting the $i$-th and $(i+1)$-th layers. We suggest using the following expression

$$\phi(\mathbf{w}) = \sum_{i=1}^{n_l-1} \sum_{l=1}^{cl_i} \left\| \mathbf{w}_l^{(i)} \right\|, \tag{3}$$

where $n_l$ is the number of layers, $\mathbf{w}_l^{(i)}$ is the $l-$th column vector of matrix $\mathbf{W}_i$, $cl_i$ is the number of column vectors in matrix $\mathbf{W}_i$. Thus the function of the optimization problem for pruning the hidden layer can be formulated as follows

$$E = \frac{1}{2} \sum_j \left\| Output^j - Target^j \right\|^2 + \lambda \sum_{i=1}^{n_l-1} \sum_{l=1}^{cl_i} \left\| \mathbf{w}_l^{(i)} \right\|, \tag{4}$$

where $\|\cdot\|$ is the $l_2$ norm (Euclidean). The above penalty term form is identical to the counterpart in Group Lasso, namely, the $l_1$-$l_2$ norm penalty, where the norm of the weight vectors, $\|\mathbf{w}_l^{(i)}\|$, is one of the components of the whole weight matrices.

It is obvious that the penalty term of the above cost function (4) is not differentiable at the origin. This may lead to difficulties for both theoretical analysis and numerical simulations, especially when the norm of weight vector is very close to zero, because the gradients of the objective function are prerequisite for common BP networks [23]. Much attention has recently been attracted on how to efficiently solve this problem [4, 33, 5, 12].

One of the popular effective solutions is to use the smoothing approximate techniques for solving the nonsmooth optimization problems [20, 13, 14, 6].

By using the Clarke generalized gradient of the objective functions, a generalized nonlinear programming circuit was introduced in the framework of nonsmooth analysis and the theory of differential inclusions [20]. To solve the nonsmooth and nonconvex optimization problems in image restorations, an improved smoothing nonlinear conjugate gradient method was suggested in [14]. In [6], a smoothing quadratic regularization algorithm was then developed for solving a class of nonsmooth, nonconvex minimization problems, which has been widely used in statistics and sparse reconstruction.

Two recent papers focused on nonconvex and nonsmooth penalization methods of neural networks with smoothing approximate techniques [7, 8]. By adopting smooth techniques, a continuous network was established for solving a non-Lipschitz optimization problem in [7]. Under the bounded level set condition of the initial point, it was demonstrated that the uniform boundedness of the solutions and the global convergence of the proposed smoothing neural network. A novel smooth neural network was presented in [8] that finds a Clarke stationary point of the non-smooth constrained optimization problem. Some more attractive features of this study include that it relaxed the restriction of the initial point to be in a feasible set and can exactly specify the prior penalty parameters of the smooth neural network.

Inspired by the above smoothing approximate techniques, we replace the original penalty term ($l_1$-$l_2$ norm) in (4) with smoothing ones. Consequently, it is beneficial for both conquering the numerical oscillations and avoiding the difficulties in theoretical analysis. One of the main topics of this paper is to fill the gap and compare the convergence for the original non-smooth methods and its smoothing counterparts.

According to different orders of sampling, there are mainly three incremental learning approaches [25]: online learning (completely stochastic order), almost-cyclic learning (special stochastic order) and cyclic learning (fixed order). To be specific, in this paper we only implement the cyclic mode.

The existing convergence results for online gradient method are mostly asymptotic convergence with a probabilistic nature since the sampling order is completely random [10, 43, 56]. The deterministic convergence of almost-cyclic and cyclic learning has been presented with a similar proof in [49, 55, 48, 45], separately. We note that the almost-cyclic learning of BP neural networks performs numerically better than the cyclic learning algorithm since the stochastic nature survives in the almost-cyclic training process [32].

The aim of this paper is to present a comprehensive study on the weak and strong convergence of the proposed smooth neural networks with cyclic learning. We particularly show that the gradient of the cost function with respect to weight vectors can approach zero and the weight updating sequence can go to a fixed point, respectively. Moreover, the rigorous proofs will be provided to ensure the convergence consistency between the smoothed neural networks and the original neural networks in terms of the Clarke differential theory.

The main contributions of this paper are as follows:

1) *A novel penalty term has been presented as a part of the cost function for BP network incremental training, which effectively prunes the connections among neurons at group level.*

We note that the penalty term borrows the idea from Group Lasso method which is beneficial to eliminating the unnecessary weights at group level. This essentially stems from the fact that the penalty term is the summation of the $l_2$ norm (not squared) of the weight vectors, i.e., all of the weights connecting with the same neuron. The simulations in Section 5 shows the better pruning performance of the proposed algorithm than those of the common BP neural networks and BP algorithm with weight decay penalty (WDBP).

2) *Smoothing techniques have been applied to approximate the penalty term of the cost function instead of using the original nonsmooth penalty.*

It is easy to know that the original proposed penalty term is non-differentiable at the origin which may lead to the numerical oscillations and result in the obstacle on theoretical analysis. The proposed smoothing techniques encourage the smooth weight updates and show sparse properties. In addition, we provide an effective way of setting suitable smoothing parameters in the training process to bridge the gap between the convergence analysis based on the two different cost functions.

3) *The weak and strong convergence of the proposed algorithm with smoothing approximation have been proved under mild assumptions on learning parameters.*

We show that the weak convergence indicates that the norm of the gradients of the smooth cost function goes to zero, and the strong convergence implies that the weight sequence tends to a fixed point (accumulation point).

4) *By selecting reasonable smoothing parameters, the proved weak and strong convergence for the proposed smoothing networks have been shown to be consistent with those of the original networks.*

For weak convergence, we prove that any accumulation point of weight sequence is both the stationary point of the smoothed neural networks and the Clarke stationary point of the original neural networks. Corresponding to the strong convergence, a restricted proof shows that the weight updating sequences converge to the same unique point as the iteration goes to infinity.

The rest of this paper is organized as follows. In the next section, we present a smoothing approximate technique to induce the new training algorithm for BP neural networks based on the use of smoothing $l_1 - l_2$ norm penalty. In Section 3, we introduce the definitions of Clarke differential and present the main convergence results. The rigorous proof of the results is provided in Section 4. In Section 6, we conclude the research with some useful remarks.

## 2. Algorithm Description

We consider a feedforward neural network with three layers. The numbers of neurons for the input, hidden and output layers are $p$, $n$ and $q$, respectively. Suppose that the training sample set is $\{\mathbf{x}^j, \mathbf{y}^j\}_{j=0}^{J-1} \subset \mathbb{R}^p \times \mathbb{R}^q$, where $\mathbf{x}^j$ and $\mathbf{y}^j$ are the input and the corresponding ideal output of the $j$-th sample, respectively. Let $\mathbf{V} = (v_{ij})_{n\times p}$ be the weight matrix connecting the input and hidden layers, and write $\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{ip})$ for $i = 1, 2, ..., n$. The weight matrix connecting the hidden and output layers is denoted by $\mathbf{U} = (u_{ki})_{q\times n}$. Let $\mathbf{u}_{r_k} = (u_{k1}, u_{k2}, ..., u_{kn}) \in \mathbb{R}^n$ and $\mathbf{u}_{c_i} = (u_{1i}, u_{2i}, ..., u_{qi})^T \in \mathbb{R}^q$ be the $k-th$ row vector and $i-th$ column vector of weight matrix $\mathbf{U}$, where $k = 1, 2, ..., q$ and $i = 1, 2, ..., n$. To simplify the presentation, we combine the weight matrix $\mathbf{U}$ and $\mathbf{V}$, and write $\mathbf{w} = (\mathbf{u}_{r_1}, ..., \mathbf{u}_{r_q}, \mathbf{v}_1, ..., \mathbf{v}_n) \in \mathbb{R}^{n(p+q)}$. Let $g$, $f$ be the given activation functions for the hidden and output layers, respectively.

For convenience, we introduce the following vector valued functions

$$G(\mathbf{z}) = (g(z_1), g(z_2), ..., g(z_n))^T, \forall \mathbf{z} \in \mathbb{R}^n, \tag{5}$$

and

$$\begin{aligned} &F(\mathbf{U}G(\mathbf{V}\mathbf{x}^j)) \\ &= \left(f(\mathbf{u}_{r_1} \cdot G(\mathbf{V}\mathbf{x}^j)), \cdots, f(\mathbf{u}_{r_q} \cdot G(\mathbf{V}\mathbf{x}^j))\right)^T. \end{aligned} \tag{6}$$

For any given input $\mathbf{x} \in \mathbb{R}^p$, the output of the hidden neurons is $G(\mathbf{V}\mathbf{x})$.

For any fixed weights $\mathbf{w}$, the error of the neural networks is defined as

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\sum_{j=0}^{J-1}\left( \left\|F\left(\mathbf{U}G\left(\mathbf{V}\mathbf{x}^j\right)\right)-\mathbf{y}^j\right\|^2 + \lambda\left(\sum_{k=1}^{q}\|\mathbf{u}_{r_k}\| + \sum_{i=1}^{n}\|\mathbf{v}_i\|\right)\right) \\ &= \sum_{j=0}^{J-1} F_j\left(\mathbf{U}G\left(\mathbf{V}\mathbf{x}^j\right)\right) + \frac{J\lambda}{2}\left(\sum_{k=1}^{q}\|\mathbf{u}_{r_k}\| + \sum_{i=1}^{n}\|\mathbf{v}_i\|\right), \end{aligned} \tag{7}$$

where $F_j\left(\mathbf{U}G\left(\mathbf{V}\mathbf{x}^j\right)\right) = \frac{1}{2}\left(\mathbf{y}^j - F\left(\mathbf{U}G\left(\mathbf{V}\mathbf{x}^j\right)\right)\right)^2$, $j = 0, 1, ..., J-1$. The gradients of the error function with respect to $\mathbf{u}_{r_k}$ and $\mathbf{v}_i$ (when $\mathbf{u}_{r_k} \neq 0$ and $\mathbf{v}_i \neq 0$) are, separately, given by

$$\begin{aligned} E_{\mathbf{u}_{r_k}} = &\sum_{j=0}^{J-1}\left(f\left(\mathbf{u}_{r_k}\cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right) - y_k^j\right) \\ &\times f'\left(\mathbf{u}_{r_k}\cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right) G\left(\mathbf{V}\mathbf{x}^j\right)^T + J\lambda\frac{\mathbf{u}_{r_k}}{\|\mathbf{u}_{r_k}\|}, \end{aligned} \tag{8}$$

$$\begin{aligned} E_{\mathbf{v}_i} = &\sum_{j=0}^{J-1}\sum_{k=1}^{q}\left(f\left(\mathbf{u}_{r_k}\cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right) - y_k^j\right) \\ &\times f'\left(\mathbf{u}_{r_k}\cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right) u_{ki} g'\left(\mathbf{v}_i\cdot\mathbf{x}^j\right)(\mathbf{x}^j)^T + J\lambda\frac{\mathbf{v}_i}{\|\ \mathbf{v}_i\ \|}. \end{aligned} \tag{9}$$

We denote that

$$E_{\mathbf{w}} = \left(E_{\mathbf{u}_{r_1}}, E_{\mathbf{u}_{r_2}}, ..., E_{\mathbf{u}_{r_q}}, E_{\mathbf{v}_1}, E_{\mathbf{v}_2}, ..., E_{\mathbf{v}_n}\right) \tag{10}$$

We note that the group lasso penalty in (7) is an intermediate case of the $l_1$ norm penalty and the squared $l_2$ penalty. From the mathematical expression, it distinguishes the different weight vectors, however, it equally evaluates the components of the same weight vector. This is the essential reason that the built learning system may prune the weights at a group level.

It is easy to find that (7) is non-differentiable at the origin, which is prone to oscillate the numerical simulations. To solve the problems caused by the nonsmoothness, we will apply smoothing approximation method in this paper. There are many different smooth functions that can be used to approximate the $l_2$ norm in the training process of BP neural networks. For any finite dimensional vector $\mathbf{z}$ and a fixed positive constant $\alpha$, we can define a smoothing function of $\|\mathbf{z}\|$ as follows:

$$h(\mathbf{z},\alpha)=\begin{cases}\|\mathbf{z}\|, & \|\mathbf{z}\|>\alpha.\\ \frac{\|\mathbf{z}\|^2}{2\alpha}+\frac{\alpha}{2}, & \|\mathbf{z}\|\le\alpha.\end{cases} \tag{11}$$

We notice that this function approximates $h(\mathbf{z})=\|\mathbf{z}\|$ closer and closer as the parameter $\alpha$ approaches to zero. Therefore, we will apply (11) to approximate the $l_2$ norm of the penalty term in (7). In addition, the gradient of $h(\mathbf{z},\alpha)$ with respect to vector $\mathbf{z}$ is

$$\nabla_{\mathbf{z}}h(\mathbf{z},\alpha)=\begin{cases}\frac{\mathbf{z}}{\|\mathbf{z}\|}, & \|\mathbf{z}\|>\alpha\\ \frac{\mathbf{z}}{\alpha}, & \|\mathbf{z}\|\le\alpha\end{cases} \tag{12}$$

The error function (7) can then be rewritten as

$$\begin{aligned}\widetilde{E}(\mathbf{w})=&\frac{1}{2}\sum_{j=0}^{J-1}\left\|F(\mathbf{U}G(\mathbf{V}\mathbf{x}^j))-\mathbf{y}^j\right\|^2\\ &+\frac{J\lambda}{2}\left(\sum_{k=1}^{q}h(\mathbf{u}_{r_k},\alpha)+\sum_{i=1}^{n}h(\mathbf{v}_i,\alpha)\right).\end{aligned} \tag{13}$$

Based on (13), we construct a smoothing Group Lasso BP network (SGLBP) as follows. Starting from an arbitrary initial guess $\mathbf{w}^0$, we proceed to refine the weight sequence iteratively by the following formulae

$$\mathbf{u}_{r_k}^{mJ+j+1}=\mathbf{u}_{r_k}^{mJ+j}-\eta_m\nabla_j\mathbf{u}_{r_k}^{mJ+j} \tag{14}$$

$$\mathbf{v}_i^{mJ+j+1}=\mathbf{v}_i^{mJ+j}-\eta_m\nabla_j\mathbf{v}_i^{mJ+j}, \tag{15}$$

where $\eta_m>0$ is the m-th cycle learning rate,

$$\nabla_j\mathbf{u}_{r_k}^{mJ+j}=H^{m,j,j}G^{m,j,j}+\lambda J\nabla_{\mathbf{u}_{r_k}}h\left(\mathbf{u}_{r_k}^{mJ+j},\alpha_m\right) \tag{16}$$

$$\begin{aligned}\nabla_j\mathbf{v}_i^{mJ+j}=&\sum_{k=1}^{q}H^{m,j,j}u_{ki}^{mJ+j}g'\left(\mathbf{v}_i^{mJ+j}\mathbf{x}^j\right)(\mathbf{x}^j)^T\\ &+\lambda J\nabla_{\mathbf{v}_i}h\left(\mathbf{v}_i^{mJ+j},\alpha_m\right),\end{aligned} \tag{17}$$

$$H^{m,l,j} \triangleq (f\left(\mathbf{u}_{r_k}^{mJ+l} \cdot G^{m,l,j}\right) - y_k^j) f'(\mathbf{u}_{r_k}^{mJ+l} \cdot G^{m,l,j}) \tag{18}$$

$$G^{m,l,j} = G\left(\mathbf{V}^{mJ+l}\mathbf{x}^j\right)^T \tag{19}$$

and,

$$\alpha_m = \frac{1}{m^\beta}, m \in \mathbb{N}, \tag{20}$$

where $0 < \beta < \frac{1}{3}$ is a positive constant, $l, j = 0, 1, \cdots, J-1$. Then the weight vectors are updated by

$$\mathbf{w}^{mJ+j+1} = \mathbf{w}^{mJ+j} - \eta_m \nabla_j \mathbf{w}^{mJ+j} \tag{21}$$

## 3. Main Results

In this section, we state the related definitions and main results of the present paper. These definitions are referred to [42].

**Definition 3.1.** *The directional derivative of $f$ at $\mathbf{x}$ in the direction $\mathbf{d}$ is*

$$f'(\mathbf{x}; \mathbf{d}) = \lim_{t \downarrow 0} \frac{f(\mathbf{x} + t\mathbf{d}) - f(\mathbf{x})}{t}. \tag{22}$$

**Definition 3.2.** *Let $\mathbf{X}$ be a Banach space, $f$ be a locally Lipschitz continuous function at $\mathbf{x}$, and let $\mathbf{d}$ be any vector in $\mathbf{X}$. The generalized directional derivative of $f$ at $\mathbf{x}$ in the direction $\mathbf{d}$ is*

$$f^o(\mathbf{x}; \mathbf{d}) = \limsup_{\substack{t \downarrow 0 \\ \mathbf{y} \to \mathbf{x}}} \frac{f(\mathbf{y} + t\mathbf{d}) - f(\mathbf{y})}{t}, \tag{23}$$

*where $\mathbf{y}$ is a vector in $\mathbf{X}$ and $t$ is a positive scalar, and $t \downarrow 0$ denotes that $t$ tends to zero monotonically and downward.*

**Definition 3.3.** *Let $f(\mathbf{x})$ be locally Lipschitz at $\mathbf{x}$. Then we say that the generalized differential (or Clarke differential) of $f$ at $\mathbf{x}$ is the set*

$$\partial f(\mathbf{x}) = \{\xi \in \mathbf{X}^* \mid f^o(\mathbf{x}; \mathbf{d}) \geq \langle \xi, \mathbf{d} \rangle \, \forall \mathbf{d} \in \mathbf{X}\} \tag{24}$$

*The $\xi$ is said to be a generalized gradient of $f$ at $\mathbf{x}$.*

The norm $\|\xi\|_*$ in conjugate space $\mathbf{X}$ is defined by

$$\|\xi\| = \sup\{\langle \xi, d \rangle \ : d \in \mathbf{X}, \|d\| \leq 1\}$$

**Definition 3.4.** *A point $\mathbf{x}^*$ is called a Clarke stationary point of $f$ if for all $\mathbf{d}$,*

$$f^o(\mathbf{x}^*; d) \geq 0, \tag{25}$$

*i.e.,*$0 \in \partial f(\mathbf{x}^*)$.

To analyze the convergence of the weight sequence (21) in the training we make the following assumptions

(A1) The activation functions $f$ and $g$ are such that $f \in \mathbb{C}^1(\mathbb{R})$ and $g \in \mathbb{C}^1(\mathbb{R})$, $f$, $g$, $f'$ and $g'$ are uniformly bounded on $\mathbb{R}$. And $f'$ and $g'$ are local Lipschitz functions.

(A2) The learning rate $\eta_m$ satisfies that $\eta_m = O(\frac{1}{m^\tau})$, where $\frac{2}{3} \leq \tau \leq 1$.

(A3) All of Clarke stationary points of the error function (7) are isolated.

(A4) There is a bounded open set $\Omega \subset \mathbb{R}^{n(p+q)}$ such that $\{\mathbf{w}^m\} \subset \Omega$ $(m \in \mathbb{N})$.

Our main results are as follows:

**Theorem 3.1.** *(**Weak Convergence**) Suppose that the assumptions (A1), (A2) and (A4) are valid. Then the weight sequence $\{\mathbf{w}^m\}$ generated by* (14) *and* (15) *is weakly convergent in the sense that*

$$\lim_{m\to\infty} \|\widetilde{E}_{\mathbf{w}}(\mathbf{w}^m)\| = 0, \quad m \in \mathbb{N}, \tag{26}$$

*where $\widetilde{E}_{\mathbf{w}}(\mathbf{w}^m)$ represents the gradient of $\widetilde{E}(\mathbf{w}^m)$ with respect to $\mathbf{w}$. In addition, we have that*

$$0 \in \partial E(\mathbf{w}^m), \quad m \in \mathbb{N}. \tag{27}$$

**Theorem 3.2.** *(**Strong Convergence**) If assumptions (A1)-(A4) are valid, then there holds the strong convergence: $\exists\ \mathbf{w}^* \in \mathbb{R}^{n(p+q)}$ such that*

$$\lim_{m\to\infty} \mathbf{w}^m = \mathbf{w}^*. \tag{28}$$

**Theorem 3.3.** *(**Consistency**) If the assumptions $(A1) - (A4)$ hold, then the smooth error function $\widetilde{E}(\mathbf{w}^m)$ and the original error function $E(\mathbf{w}^m)$ converge to the same value as $m$ goes to infinity, that is,*

$$\lim_{m\to\infty} \widetilde{E}(\mathbf{w}^m) = \lim_{m\to\infty} E(\mathbf{w}^m) = E(\mathbf{w}^*). \tag{29}$$

**Remark:** We show that the above assumptions $(A1) - (A4)$ are sufficient but not necessary conditions for the convergence results of **Theorem** 3.1$-$**Theorem** 3.3. Stochastic gradient descent (SGD) method is one of the most common used strategies in dealing with large -scale machine learning problems. Actually, the training algorithm proposed in this paper is one of the specific cases of SGD, the fed sequence of the samples are with fixed order in the total training procedure. We note that the practical strategies of learning rates in [51, 52, 53] are of the special cases of presented assumption (A2).

To our best knowledge, it firstly shows the strict convergence analysis for incremental gradient neural networks with a constant learning rate in [54]. However, its theoretical analysis is limited for two-layer neural networks, which can not simply extend to a multilayer network. The convergence analyses of our work are beneficial to expanding the results of [54] to more general cases in the future.

## 4. Proofs

Some necessary lemmas for the proofs are given as follows.

**Lemma 4.1.** *Suppose that* $f(\mathbf{x}) = \|\mathbf{x}\|, x \in \mathbb{R}^n$, *and*

$$\widetilde{f}(\mathbf{x}, \alpha) = \begin{cases} \|\mathbf{x}\|, & \text{if } \|\mathbf{x}\| > \alpha, \\ \frac{\|\mathbf{x}\|^2}{2\alpha} + \frac{\alpha}{2}, & \text{if } \|\mathbf{x}\| \le \alpha. \end{cases} \tag{30}$$

*Then* $\lim\limits_{\alpha \downarrow 0} \partial \widetilde{f}(\mathbf{x}, \alpha) \subseteq \partial f(\mathbf{x})$.

*Proof.* We consider the following two cases to verify the result.
Case (I). If $n = 1$, the function $f(x)$ corresponds to the simple absolute value function, that is, $f(x) = |x|$. It is clear that the function $f$ is Lipschitz in terms of the triangle inequality. If $x > 0$, we get the following by Definition 3.2

$$f^o(x; d) = \limsup_{\substack{y \to x \\ t \downarrow 0}} \frac{y + td - y}{t} = d. \tag{31}$$

According to Definition 3.3,

$$\partial f(x) = \{\xi | d \ge \xi d, \forall d \in \mathbb{R}\} \tag{32}$$

includes the singleton $\{1\}$.
Similarly, we obtain that

$$\partial f(x) = \{-1\}\,, \text{ if } x < 0. \tag{33}$$

For the case $x = 0$, we can calculate

$$f^o(0; d) = \begin{cases} d, & \text{if} \quad d \geq 0, \\ -d, & \text{if} \quad d \leq 0. \end{cases} \tag{34}$$

We then immediately have that $\partial f(0) = [-1, 1]$. Therefore, we conclude that

$$\partial f(x) = \begin{cases} 1, & \text{if} \quad x > 0, \\ -1, & \text{if} \quad x < 0, \\ [-1, 1], & \text{if} \quad x = 0. \end{cases} \tag{35}$$

We note that the derivative of (30) with respect to $x$ can be obtained as following

$$\partial \widetilde{f}(x, \alpha) = \begin{cases} \frac{x}{\|x\|}, & \text{if } \|x\| > \alpha, \\ \frac{x}{\alpha}, & \text{if } \|x\| \leq \alpha. \end{cases} \tag{36}$$

Then we have that

$$\left| \lim_{\alpha \downarrow 0} \partial \widetilde{f}(x, \alpha) \right| \leq 1. \tag{37}$$

This then implies that $\lim\limits_{\alpha \downarrow 0} \partial \widetilde{f}(x, \alpha) \subseteq \partial f(x)$.
Case (II). When $n > 1$, $f(\mathbf{x}) = \|\mathbf{x}\|$. We let $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$, then the Clarke differential is as follows

$$\partial f(\mathbf{x}) = \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|}, & \text{if } \|\mathbf{x}\| > 0, \\ \{\xi \in \mathbf{R}^n : \|\xi\| \leq 1\}\,, & \text{if } \|\mathbf{x}\| = 0. \end{cases} \tag{38}$$

The gradient of $\widetilde{f}(\mathbf{x}, \alpha)$ with respect to $\mathbf{x}$ can be expressed with

$$\partial \widetilde{f}(\mathbf{x}, \alpha) = \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|}, & \text{if } \|\mathbf{x}\| > \alpha, \\ \frac{\mathbf{x}}{\alpha}, & \text{if } \|\mathbf{x}\| \leq \alpha. \end{cases} \tag{39}$$

It is clear that $\lim\limits_{\alpha \downarrow 0} \partial \widetilde{f}(\mathbf{x}, \alpha) \subseteq \partial f(\mathbf{x})$. This then completes the proof. $\square$

**Lemma 4.2.** *Suppose that the learning rate $\eta_m > 0$ satisfies that the assumption (A2). And the sequence $\{a_m\}$ $(m \in \mathbb{N})$ satisfies that $a_m > 0$, $\sum\limits_{m=0}^{\infty} \eta_m a_m^{\gamma} < \infty$ and $\lim_{m\to\infty} |a_{m+1} - a_m| = 0$ for a fixed positive constant $\gamma$. Then we have*

$$\lim_{m\to\infty} a_m = 0. \tag{40}$$

*The details of the proof has been done in [48], and omitted here.*

**Lemma 4.3.** *Suppose that the assumptions (A1), (A2) and (A4) are valid. Then the weight sequence $\{\mathbf{w}^m\}$ generated by* (14) *and* (15) *satisfies the following weak convergence*

$$\lim_{m\to\infty} \|\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{m+1}) - \widetilde{E}_{\mathbf{w}}(\mathbf{w}^m)\| = 0. \tag{41}$$

*Proof.* Due to the similarity, we just prove the result of $\lim\limits_{m\to\infty} \|\widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ+j})\| = 0$ and omit the proof of $\lim\limits_{m\to\infty} \|\widetilde{E}_{\mathbf{v}_i}(\mathbf{w}^m)\| = 0$.

By virtue of (13), we know that

$$\widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ+j}) = \sum_{j=0}^{J-1} H^{m,j,j} G^{m,j,j} + J\lambda \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ+j}, \alpha_m), \tag{42}$$

$$\widetilde{E}_{\mathbf{v}_i}(\mathbf{w}^{mJ+j}) = \sum_{j=0}^{J-1} \sum_{k=1}^{q} H^{m,j,j} u_{ki} g'\left(\mathbf{v}_i \cdot \mathbf{x}^j\right)(\mathbf{x}^j)^T + J\lambda \nabla_{\mathbf{v}_i} h(\mathbf{v}_i^{mJ+j}, \alpha_m), \tag{43}$$

which yields

$$\begin{aligned}
&\left\|\widetilde{E}_{\mathbf{u}_{r_k}}\left(\mathbf{w}^{(m+1)J}\right) - \widetilde{E}_{\mathbf{u}_{r_k}}\left(\mathbf{w}^{mJ}\right)\right\| \\
&= \| \sum_{j=0}^{J-1} \left(H^{m+1,0,j} G^{m+1,0,j} - H^{m,0,j} G^{m,0,j}\right) \\
&\quad + J\lambda \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m) - J\lambda \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) \| \\
&\le \sum_{j=0}^{J-1} \left\|\left(H^{m+1,0,j} G^{m+1,0,j} - H^{m,0,j} G^{m,0,j}\right)\right\| \\
&\quad + J\lambda \left\|\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m) - \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m)\right\|.
\end{aligned} \tag{44}$$

From Assumption (A4), we can see that there are $C_1 > 0$ and $C_2 > 0$ such that

$$\max_{0\le j\le J-1}\left\{\|\mathbf{x}^j\|, \|\mathbf{y}^j\|\right\} = C_1, \sup_{m\in\mathbb{N}} \|\mathbf{w}^m\| = C_2. \tag{45}$$

First of all, we know that $f$, $g$ are continuously differentiable. And $f'$ and $g'$ are Lipschitz continuous and uniformly bounded on $\mathbb{R}$. Thus, there exist three positive constants $C_3, C_4$ and $C_5$, such that

$$|f\,(x)| \le C_3, \forall x \in \mathbb{R}; \tag{46}$$

$$\|G(\mathbf{z})\| \le C_4, \forall \mathbf{z} \in \mathbb{R}^n; \tag{47}$$

$$|f'\,(x)| \le C_5, \forall x \in \mathbb{R}; \tag{48}$$

$$|f'\,(x) - f'\,(y)| \le L_{f'}|x-y|, \forall x, y \in \mathbb{R}. \tag{49}$$

where $L_{f'}$ represents the Lipschitz constant of $f'$.

$$\begin{aligned}\left\|\nabla_j \mathbf{u}_{r_k}^{mJ+j}\right\| &= \left\|H^{m,j,j}G^{m,j,j} + \lambda\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ+j}, \alpha_m)\right\| \\ &\le (\sup_{x\in\mathbb{R}} |f(x)| + |y_k^j|)C_5C_4 + \lambda \\ &\le C_6,\end{aligned} \tag{50}$$

Similarly, $\left\|\nabla_j \mathbf{v}_i^{mJ+j}\right\| \le C_7$, for some $C_7 > 0$. Now let's consider the first part of (44):

$$\left\|H^{m+1,0,j}G^{m+1,0,j} - H^{m,0,j}G^{m,0,j}\right\| \le (\Gamma_1 + \Gamma_2 + \Gamma_3) \tag{51}$$

where

$$\begin{aligned}\Gamma_1 =& \left|f\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j}\right) - f\left(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right)\right| \\ &\times \left|f'\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j}\right)\right| \left\|G^{m+1,0,j}\right\|,\end{aligned} \tag{52}$$

$$\begin{aligned}\Gamma_2 =& \left|f'\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j}\right) - f'\left(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right)\right| \\ &\times \left|f\left(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right) - y_k^j\right| \left\|G^{m+1,0,j}\right\|,\end{aligned} \tag{53}$$

$$\begin{aligned}\Gamma_3 =& \left\|G^{m+1,0,j} - G^{m,0,j}\right\| \left|f(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}) - y_k^j\right| \\ &\times \left|f'(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j})\right|.\end{aligned} \tag{54}$$

Then by Newton-Leibniz formula, we find

$$
\begin{aligned}
\Gamma_1 = & \left| \int_{\mathbf{u}^{mJ} G^{m,0,j}}^{\mathbf{u}_{r_k}^{(m+1)J} G^{m+1,0,j}} f'(x) dx \right| \\
& \times \left| f'\left( \mathbf{u}_{r_k}^{(m+1)J} \cdot G^{m+1,0,j} \right) \right| \left\| G^{m+1,0,j} \right\| \\
& \le C_5 | \mathbf{u}_{r_k}^{(m+1)J} \cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ} \cdot G^{m,0,j} | C_5 C_4 \\
& \le C_8 \Big( \| \mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ} \| \| G^{m+1,0,j} \| \\
& \quad + \| \mathbf{u}_{r_k}^{mJ} \| \| G^{m+1,0,j} - G^{m,0,j} \| \Big) \\
& \le C_8 \Big( C_4 \sum_{j=0}^{J-1} \eta_m \| \nabla_j \mathbf{u}_{r_k}^{mJ+j} \| \\
& \quad + \| \mathbf{u}_{r_k}^{mJ} \| \sup_{x \in \mathbb{R}} |g'(x)| \, \| \mathbf{x}^j \| \sum_{i=1}^{n} \sum_{j=0}^{J-1} \eta_m \| \nabla_j \mathbf{v}_i^{mJ+j} \| \Big) \\
& \le \eta_m \left( J C_4 C_6 C_8 + n J C_1 C_2 \sup_{x \in \mathbb{R}} |g'(x)| \, C_7 C_8 \right) \\
& \le C_9 \eta_m,
\end{aligned}
\tag{55}
$$

where $C_8 = C_4 C_5^2$, $C_9 = J C_4 C_6 C_8 + n J C_1 C_2 \sup_{x \in \mathbb{R}} |g'(x)| \, C_7 C_8)$. By the same way, we can obtain the following by (49)

$$
\begin{aligned}
\Gamma_2 \le & L_{f'} \left| \mathbf{u}_{r_k}^{(m+1)J} \cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ} \cdot G^{m,0,j} \right| \\
& \times \left( \sup_{x \in \mathbb{R}} |f(x)| + |y_k^j| \right) C_4 \\
& \le L_{f'} \left| \mathbf{u}_{r_k}^{(m+1)J} \cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ} \cdot G^{m,0,j} \right| (C_3 + C_1) C_4 \\
& \le C_{10} \eta_m,
\end{aligned}
\tag{56}
$$

where $L_{f'}$ denotes the Lipschitz coefficient of $f'$, and

$C_{10} = L_{f'}(JC_4C_6 + nJC_1C_2 \sup_{x\in\mathbb{R}} |g'(x)|\, C_7)(C_1 + C_3)C_4$.

$$\begin{aligned}\Gamma_3 &\le \sup_{x\in\mathbb{R}} |g'(x)| \|\mathbf{x}^j\| \sum_{i=1}^{n} \sum_{j=0}^{J-1} \|\eta_m \nabla_j \mathbf{v}_i^{mJ+j}\| \\ &\quad \times (\sup_{x\in\mathbb{R}} |f(x)| + |y_k^j|)C_5 \\ &\le \sup_{x\in\mathbb{R}} |g'(x)| C_1 nJ\eta_m C_7(C_3 + C_1)C_5 \\ &\le C_{11}\eta_m,\end{aligned} \tag{57}$$

where $C_{11} = \sup_{x\in\mathbb{R}} |g'(x)| C_1 nJC_7(C_3 + C_1)C_5$.

Now we study the second part of (44) with the following four cases:

$$\begin{cases} \frac{\mathbf{u}_{r_k}^{(m+1)J}}{\|\mathbf{u}_{r_k}^{(m+1)J}\|} - \frac{\mathbf{u}_{r_k}^{mJ}}{\|\mathbf{u}_{r_k}^{mJ}\|}, \text{ for } \|\mathbf{u}_{r_k}^{(m+1)J}\| > \alpha_{m+1}, \|\mathbf{u}_{r_k}^{mJ}\| > \alpha_m. \\ \frac{\mathbf{u}_{r_k}^{(m+1)J}}{\|\mathbf{u}_{r_k}^{(m+1)J}\|} - \frac{\mathbf{u}_{r_k}^{mJ}}{\alpha_m}, \quad \text{for } \|\mathbf{u}_{r_k}^{(m+1)J}\| > \alpha_{m+1}, \|\mathbf{u}_{r_k}^{mJ}\| \le \alpha_m. \\ \frac{\mathbf{u}_{r_k}^{(m+1)J}}{\alpha_{m+1}} - \frac{\mathbf{u}_{r_k}^{mJ}}{\|\mathbf{u}_{r_k}^{mJ}\|}, \quad \text{for } \|\mathbf{u}_{r_k}^{(m+1)J}\| \le \alpha_{m+1}, \|\mathbf{u}_{r_k}^{mJ}\| > \alpha_m. \\ \frac{\mathbf{u}_{r_k}^{(m+1)J}}{\alpha_{m+1}} - \frac{\mathbf{u}_{r_k}^{mJ}}{\alpha_m}, \quad \text{for } \|\mathbf{u}_{r_k}^{(m+1)J}\| \le \alpha_{m+1}, \|\mathbf{u}_{r_k}^{mJ}\| \le \alpha_m. \end{cases} \tag{58}$$

We just study the first case because of the similarity of analyzing the other cases. From (A2), $\eta_m = o(\alpha_m)$.

$$\begin{aligned}&\left| \frac{\mathbf{u}_{r_k}^{(m+1)J}}{\left\|\mathbf{u}_{r_k}^{(m+1)J}\right\|} - \frac{\mathbf{u}_{r_k}^{mJ}}{\|\mathbf{u}_{r_k}^{mJ}\|} \right| \\ &\le \frac{\left\|\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right\|}{\left\|\mathbf{u}_{r_k}^{(m+1)J}\right\|} + \|\mathbf{u}_{r_k}^{mJ}\| \frac{\left\|\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right\|}{\left\|\mathbf{u}_{r_k}^{(m+1)J}\right\| \|\mathbf{u}_{r_k}^{mJ}\|} \\ &= 2\frac{\left\|\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right\|}{\left\|\mathbf{u}_{r_k}^{(m+1)J}\right\|} \\ &\le 2\frac{\left\|\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right\|}{\alpha_m} \le \frac{C_{12}\eta_m}{\alpha_m},\end{aligned} \tag{59}$$

where $C_{12} = 2JC_6$. Thus we obtain that

$$J\lambda \|\nabla h_{\mathbf{u}_{r_k}}(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m) - \nabla h_{\mathbf{u}_{r_k}}(\mathbf{u}_{r_k}^{mJ}, \alpha_m)\| \leq C_{13}\frac{\eta_m}{\alpha_m}. \tag{60}$$

where $C_{13} = J\lambda C_{12}$.

In terms of (44), (51), (55), (56), (57) and (60), we conclude that

$$\begin{aligned} &\left\| \widetilde{E}_{\mathbf{u}_{r_k}}\left(\mathbf{w}^{(m+1)J}\right) - \widetilde{E}_{\mathbf{u}_{r_k}}\left(\mathbf{w}^{mJ}\right) \right\| \\ &\leq JC_9\eta_m + JC_{10}\eta_m + JC_{11}\eta_m + C_{13}\frac{\eta_m}{\alpha_m} \\ &\leq C_{14}(3\eta_m + \frac{\eta_m}{\alpha_m}) \to 0, (m \to \infty) \end{aligned} \tag{61}$$

where $C_{14} = \max\{JC_9, JC_{10}, JC_{11}, C_{13}\}$. □

**Lemma 4.4.** *Let $q(x)$ be a function defined on a bounded closed interval [a, b], such that $q'(x)$ is Lipschitz continuous with Lipschitz constant $K > 0$. Then $q(x)$ is differentiable almost everywhere in [a, b] and*

$$|q''(x)| \leq K, a.e.[a, b]. \tag{62}$$

*Moreover, there exists a constant $C > 0$ such that*

$$q(x) \leq q(x_0) + q'(x_0)(x - x_0) + C(x - x_0)^2, \tag{63}$$

*where $x_0, x \in [a, b]$.*

*The details are referred to [48] for interested readers.*

The next lemma plays an essential role in assuring the weak convergence theorem 3.1. It also reveals an almost monotonicity of the smoothing error function during training procedure. Certainly, the rigorous proof of this lemma is a little complicated in which the Taylor expansion and inequality tricks have been frequently used.

**Lemma 4.5.** *Let the sequence $\left\{\mathbf{w}^{mJ+j}\right\}$ be generated by* (14) *and* (15)*. Under assumptions (A1), (A2) and (A4), there holds*

$$\begin{aligned} \widetilde{E}(\mathbf{w}^{(m+1)J}) \leq &\widetilde{E}(\mathbf{w}^{mJ}) - \eta_m \left\| \widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ}) \right\|^2 + \delta_m \\ &+ J(n+q)\frac{\alpha_m - \alpha_{m+1}}{2}. \end{aligned} \tag{64}$$

*Proof.* By virtue of (13), we have

$$\begin{aligned}\widetilde{E}(\mathbf{w}^{mJ+j}) &= \frac{1}{2}\sum_{j=0}^{J-1}\left\|F(\mathbf{U}^{mJ+j}G(\mathbf{V}^{mJ+j}\mathbf{x}^j)) - \mathbf{y}^j\right\|^2 \\ &\quad + J\lambda\left(\sum_{k=1}^{q} h(\mathbf{u}_{r_k}^{mJ+j}, \alpha_m) + \sum_{i=1}^{n} h(\mathbf{v}_i^{mJ+j}, \alpha_m)\right) \\ &= \frac{1}{2}\sum_{j=0}^{J-1}\sum_{k=1}^{q}\left|f(\mathbf{u}_{r_k}^{mJ+j}\cdot G(\mathbf{V}^{mJ+j}\mathbf{x}^j)) - y_k^j\right|^2 \\ &\quad + J\lambda(\sum_{k=1}^{q} h(\mathbf{u}_{r_k}^{mJ+j}, \alpha_m) + \sum_{i=1}^{n} h(\mathbf{v}_i^{mJ+j}, \alpha_m)),\end{aligned} \tag{65}$$

where $j = 0, 1, \cdots, J-1$ and $m \in \mathbb{N}$.

From Lemma 4.4, (16), (17) and assumption (A1), there is a constant $C_{15} > 0$ such that

$$\begin{aligned}&\left(f(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j}) - y_k^j\right)^2 \\ &\le \left(f(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}) - y_k^j\right)^2 \\ &\quad + 2H^{m,0,j}\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right) \\ &\quad + C_{15}\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right)^2.\end{aligned} \tag{66}$$

Let

$$\Gamma_4 = \left(f(\mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}) - y_k^j\right)^2, \tag{67}$$

$$\Gamma_5 = 2H^{m,0,j}\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right), \tag{68}$$

$$\Gamma_6 = C_{15}\left(\mathbf{u}_{r_k}^{(m+1)J}\cdot G^{m+1,0,j} - \mathbf{u}_{r_k}^{mJ}\cdot G^{m,0,j}\right)^2. \tag{69}$$

Then from (68), we find

$$\begin{aligned}\Gamma_5 = 2H^{m,0,j}\big[&(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ})\cdot G^{m,0,j} \\ &+ \mathbf{u}_{r_k}^{mJ}\cdot(G^{m+1,0,j} - G^{m,0,j}) \\ &+ (\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ})\cdot(G^{m+1,0,j} - G^{m,0,j})\big].\end{aligned} \tag{70}$$

By virtue of the integral Taylor expansion and assumption (A1), $g''(x)$ exits almost everywhere which implies

$$\begin{aligned}
&\mathbf{u}_{r_k}^{mJ} \cdot (G^{m+1,0,j} - G^{m,0,j}) \\
&= \sum_{i=1}^{n} u_{r_k,i}^{mJ} g'(\mathbf{v}_i^{mJ} \cdot \mathbf{x}^j) \left(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ}\right) \cdot \mathbf{x}^j \\
&\quad + \sum_{i=1}^{n} u_{r_k,i}^{mJ} \left[\left(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ}\right) \cdot \mathbf{x}^j\right]^2 \\
&\quad \times \int_0^1 (1-t) g''(\mathbf{v}^{mJ} \cdot \mathbf{x} + t(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ})) dt,
\end{aligned} \tag{71}$$

where $u_{r_k,i}^{mJ}$ is the $i$-th element of the weight vector $\mathbf{u}_{r_k}^{mJ}$.

$$h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_{m+1}) - h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) = \Gamma_7 + \Gamma_8, \tag{72}$$

where

$$\Gamma_7 = h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m) - h(\mathbf{u}_{r_k}^{mJ}, \alpha_m), \tag{73}$$

$$\Gamma_8 = h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_{m+1}) - h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m). \tag{74}$$

Using Taylor formula,

$$\begin{aligned}
&h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m) \\
&= h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) + \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) \cdot \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right) \\
&\quad + \frac{1}{2} \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right) \nabla\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_\xi, \alpha_\xi) \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right)^T,
\end{aligned} \tag{75}$$

where $\mathbf{u}_\xi = \theta \mathbf{u}_{r_k}^{(m+1)J} + (1-\theta)\mathbf{u}_{r_k}^{mJ}$, $\alpha_\xi = \theta\alpha_{m+1} + (1-\theta)\alpha_m$, $0 < \theta < 1$. Since $h(\mathbf{z}, \alpha)$ is smooth and continuous differential, it is easy to find that $\|\nabla\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_\xi, \alpha_\xi)\|$ is bounded by $O\left(\frac{1}{\alpha_m}\right)$. So we have

$$\begin{aligned}
h(\mathbf{u}_{r_k}^{(m+1)J}, \alpha_m) &\le h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) + \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) \cdot \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right) \\
&\quad + \frac{1}{2} \sup_{\substack{\mathbf{u}_\xi \in \Omega \\ \alpha_\xi \in \mathbb{R}}} \|\nabla\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_\xi, \alpha_\xi)\| \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right)^2 \\
&\le h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) + \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) \cdot \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right) \\
&\quad + C_{16}^m \left\|\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right\|^2.
\end{aligned} \tag{76}$$

where $C_{16}^m = \sup\left\{\frac{1}{2}\sup_{\substack{\mathbf{u}_\xi\in\Omega\\ \alpha_\xi\in\mathbb{R}}}\|\nabla\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_\xi,\alpha_\xi)\|\right\}$. It is obvious that

$$\begin{aligned}\Gamma_7 \leq& \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ},\alpha_m)\cdot\left(\mathbf{u}_{r_k}^{(m+1)J}-\mathbf{u}_{r_k}^{mJ}\right)\\ &+C_{16}^m\left\|\mathbf{u}_{r_k}^{(m+1)J}-\mathbf{u}_{r_k}^{mJ}\right\|^2.\end{aligned} \tag{77}$$

By (11), we obtain the following

$$\Gamma_8 = \begin{cases} 0, & \|\mathbf{u}_{r_k}^{(m+1)J}\| \geq \alpha_m,\\ -\frac{\left(\|\mathbf{u}_{r_k}^{(m+1)J}\|-\alpha_m\right)^2}{2\alpha_m}, & \|\mathbf{u}_{r_k}^{(m+1)J}\| \in (\alpha_{m+1},\alpha_m),\\ \gamma_m, & \|\mathbf{u}_{r_k}^{(m+1)J}\| \leq \alpha_{m+1},\end{cases} \tag{78}$$

where $\gamma_m = \frac{(\alpha_m-\alpha_{m+1})\left(\left\|\mathbf{u}_{r_k}^{(m+1)J}\right\|^2-\alpha_m\alpha_{m+1}\right)}{2\alpha_m\alpha_{m+1}}$. This implies that

$$|\Gamma_8| \leq \frac{\alpha_m-\alpha_{m+1}}{2}. \tag{79}$$

Thus, we have

$$\begin{aligned}&h(\mathbf{u}_{r_k}^{(m+1)J},\alpha_{m+1})\\ &\leq h(\mathbf{u}_{r_k}^{mJ},\alpha_m)+\nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ},\alpha_m)\cdot\left(\mathbf{u}_{r_k}^{(m+1)J}-\mathbf{u}_{r_k}^{mJ}\right)\\ &\quad+C_{16}^m\left\|\mathbf{u}_{r_k}^{(m+1)J}-\mathbf{u}_{r_k}^{mJ}\right\|^2+\frac{\alpha_m-\alpha_{m+1}}{2}.\end{aligned} \tag{80}$$

By the same way, we can find

$$\begin{aligned}&h(\mathbf{v}_i^{(m+1)J},\alpha_{m+1})\\ &\leq h(\mathbf{v}_i^{mJ},\alpha_m)+\nabla_{\mathbf{v}_i} h(\mathbf{v}_i^{mJ},\alpha_m)\cdot\left(\mathbf{v}_i^{(m+1)J}-\mathbf{v}_i^{mJ}\right)\\ &+C_{17}^m\left\|\mathbf{v}_i^{(m+1)J}-\mathbf{v}_i^{mJ}\right\|^2+\frac{\alpha_m-\alpha_{m+1}}{2}.\end{aligned} \tag{81}$$

By virtue of (14) and (15), we then obtain

$$\begin{aligned}\mathbf{u}_{r_k}^{(m+1)J}-\mathbf{u}_{r_k}^{mJ} &= \sum_{j=0}^{J-1}\eta_m\nabla_j\mathbf{u}_{r_k}^{mJ+j}\\ &= \sum_{j=0}^{J-1}\eta_m\nabla_j\mathbf{u}_{r_k}^{mJ}+\sum_{j=0}^{J-1}\eta_m\left(\nabla_j\mathbf{u}_{r_k}^{mJ+j}-\nabla_j\mathbf{u}_{r_k}^{mJ}\right),\end{aligned} \tag{82}$$

$$
\begin{aligned}
\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ} &= \sum_{j=0}^{J-1} \eta_m \nabla_j \mathbf{v}_i^{mJ+j} \\
&= \sum_{j=0}^{J-1} \eta_m \nabla_j \mathbf{v}_i^{mJ} + \sum_{j=0}^{J-1} \eta_m \left( \nabla_j \mathbf{v}_i^{mJ+j} - \nabla_j \mathbf{v}_i^{mJ} \right).
\end{aligned}
\tag{83}
$$

Summing (66) from $k = 1$ to $k = q$ and $j = 0$ to $j = J-1$, and then multiple $\frac{1}{2}$, summing (76) from $k = 1$ to $k = q$ and summing (81) from $i = 1$ to $i = n$, and also, from (16), (17), (67)-(81), (82), (83), (42), (43), we then obtain

$$
\begin{aligned}
\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{(m+1)J}) \le{}& \frac{1}{2} \sum_{j=0}^{J-1} \sum_{k=1}^{q} \Gamma_4 + \sum_{j=0}^{J-1} \sum_{k=1}^{q} H^{m,0,j} (\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}) \cdot G^{m,0,j} \\
&+ \sum_{j=0}^{J-1} \sum_{k=1}^{q} H^{m,0,j} \Bigg( \sum_{i=1}^{n} u_{r_k,i}^{mJ} g'(\mathbf{v}_i^{mJ} \cdot \mathbf{x}^j) \left( \mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ} \right) \cdot \mathbf{x}^j \\
&+ \sum_{i=1}^{n} u_{r_k,i}^{mJ} \left[ \left( \mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ} \right) \cdot \mathbf{x}^j \right]^2 \\
&\times \int_0^1 (1-t) g''(\mathbf{v}^{mJ} \cdot \mathbf{x} + t(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ})) dt \Bigg) \\
&+ \sum_{j=0}^{J-1} \sum_{k=1}^{q} H^{m,0,j} (\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}) \cdot (G^{m+1,0,j} - G^{m,0,j}) \\
&+ \sum_{j=0}^{J-1} \sum_{k=1}^{q} C_{15} \Gamma_6 + J\lambda \sum_{k=1}^{q} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) \\
&+ J\lambda \sum_{k=1}^{q} \nabla_{\mathbf{u}_{r_k}} h(\mathbf{u}_{r_k}^{mJ}, \alpha_m) \cdot \left( \mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ} \right) \\
&+ J\lambda \sum_{k=1}^{q} C_{16}^m \left\| \mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ} \right\|^2 + J\lambda \sum_{i=1}^{n} h(\mathbf{v}_i^{mJ}, \alpha_m) \\
&+ J\lambda \sum_{i=1}^{n} \nabla_{\mathbf{v}_i} h(\mathbf{v}_i^{mJ}, \alpha_m) \cdot \left( \mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ} \right) \\
&+ J\lambda \sum_{i=1}^{n} C_{17}^m \left\| \mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ} \right\|^2 + J(n+q)\lambda \frac{\alpha_m - \alpha_{m+1}}{2}
\end{aligned}
$$

$$
\begin{aligned}
=&\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ}) + \sum_{k=1}^{q} \left(\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}\right) \cdot \widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ}) \\
&+ \sum_{i=1}^{n} \left(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ}\right) \cdot \widetilde{E}_{\mathbf{v}_i}(\mathbf{w}^{mJ}) + \zeta_m \\
&+ J(n+q)\lambda \frac{\alpha_m - \alpha_{m+1}}{2} \\
=&\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ}) - \sum_{k=1}^{q} \left(\sum_{j=0}^{J-1} \eta_m \nabla_j \mathbf{u}_{r_k}^{mJ}\right) \cdot \widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ}) \\
&- \sum_{i=1}^{n} \left(\sum_{j=0}^{J-1} \eta_m \nabla_j \mathbf{v}_i^{mJ}\right) \cdot \widetilde{E}_{\mathbf{v}_i}(\mathbf{w}^{mJ}) + \delta_m \\
&+ J(n+q)\lambda \frac{\alpha_m - \alpha_{m+1}}{2} \\
=&\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ}) - \eta_m \sum_{k=1}^{q} \left\| \widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ}) \right\|^2 \\
&- \eta_m \sum_{i=1}^{n} \left\| \widetilde{E}_{\mathbf{v}_i}^2(\mathbf{w}^{mJ}) \right\|^2 + \delta_m + J(n+q)\lambda \frac{\alpha_m - \alpha_{m+1}}{2} \\
=&\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ}) - \eta_m \left\| \widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ}) \right\|^2 + \delta_m \\
&\quad + J(n+q)\lambda \frac{\alpha_m - \alpha_{m+1}}{2}, \\
\zeta_m =& \sum_{j=0}^{J-1} \sum_{k=1}^{q} H^{m,0,j} (\mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ}) \cdot (G^{m+1,0,j} - G^{m,0,j}) \\
&\quad + \sum_{j=0}^{J-1} \sum_{k=1}^{q} H^{m,0,j} \sum_{i=1}^{n} u_{r_k,i}^{mJ} \left[ \left(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ}\right) \cdot \mathbf{x}^j \right]^2 \\
&\quad \times \int_0^1 (1-t) g''(\mathbf{v}^{mJ} \cdot \mathbf{x} + t(\mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ})) dt \qquad (84) \\
&\quad + \sum_{j=0}^{J-1} \sum_{k=1}^{q} C_{15} \Gamma_6 + J \sum_{k=1}^{q} C_{16}^m \left\| \mathbf{u}_{r_k}^{(m+1)J} - \mathbf{u}_{r_k}^{mJ} \right\|^2 \\
&\quad + J \sum_{i=1}^{n} C_{17}^m \left\| \mathbf{v}_i^{(m+1)J} - \mathbf{v}_i^{mJ} \right\|^2,
\end{aligned}
$$

and

$$\begin{aligned}\delta_m =& \left(\sum_{j=0}^{J-1} \eta_m \left(\nabla_j \mathbf{u}_{r_k}^{mJ+j} - \nabla_j \mathbf{u}_{r_k}^{mJ}\right)\right) \cdot \widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ}) \\ &+ \left(\sum_{j=0}^{J-1} \eta_m \left(\nabla_j \mathbf{v}_i^{mJ+j} - \nabla_j \mathbf{v}_i^{mJ}\right)\right) \cdot \widetilde{E}_{\mathbf{v}_i}(\mathbf{w}^{mJ}) + \zeta_m.\end{aligned} \tag{85}$$

By a similar argument with (51)-(61) and (14), we can find that

$$\left\|\nabla_j \mathbf{u}_{r_k}^{mJ+j} - \nabla_j \mathbf{u}_{r_k}^{mJ}\right\| \leq C_{18}\eta_m \tag{86}$$

$$\left\|\nabla_j \mathbf{v}_i^{mJ+j} - \nabla_j \mathbf{v}_i^{mJ}\right\| \leq C_{19}\eta_m, \tag{87}$$

for some $C_{18},\ C_{19} > 0$. By virtue of (45)-(48), (42) and (43), it is easy to see that the first term of $\delta_m$ can be controlled by

$$\begin{aligned}&\left(\sum_{j=0}^{J-1} \eta_m \left(\nabla_j \mathbf{u}_{r_k}^{mJ+j} - \nabla_j \mathbf{u}_{r_k}^{mJ}\right)\right) \cdot \widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ}) \\ &\leq \eta_m C_{18}\eta_m \left((\sup_{x\in\mathbb{R}} |f(x)| - y_k^j) C_5 C_4 + \lambda\right) \leq C_{20}\eta_m^2\end{aligned} \tag{88}$$

In the same way, we can estimate the other terms of $\delta_m$ with corresponding constants $C_{21}, ..., C_{26} > 0$. Finally, the desired estimate (64) could be proved by setting $C_{27} = \sum_{v=21}^{26} C_v$. □

**Proof of Theorem 3.1.** It is easy to see that the series $\sum_{m=1}^{\infty} \frac{\alpha_m - \alpha_{m+1}}{2} < \infty$. In addition, by Lemma 4.5, we can conclude that

$$\begin{aligned}&\sum_{m=0}^{\infty} \eta_m \left\|\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{mJ})\right\|^2 \\ &= \sum_{m=0}^{\infty} \eta_m \left(\sum_{k=1}^{q} \left\|\widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ})\right\|^2 + \sum_{i=1}^{n} \left\|\widetilde{E}_{\mathbf{v}_i}(\mathbf{w}^{mJ})\right\|^2\right) \\ &< \infty,\end{aligned} \tag{89}$$

which shows that

$$\sum_{m=0}^{\infty} \eta_m \left\|\widetilde{E}_{\mathbf{u}_{r_k}}(\mathbf{w}^{mJ})\right\|^2 < \infty \tag{90}$$

From Lemma 4.2 and (61), it then follows that

$$\lim_{m\to\infty}\left\|\widetilde{E}_{\mathbf{u}_{r_k}}\left(\mathbf{w}^{mJ}\right)\right\| = 0. \tag{91}$$

Similarly we have

$$\lim_{m\to\infty}\left\|\widetilde{E}_{\mathbf{u}_{r_k}}\left(\mathbf{w}^{mJ+j}\right)\right\| = 0, \tag{92}$$

and

$$\lim_{m\to\infty}\left\|\widetilde{E}_{\mathbf{v}_i}\left(\mathbf{w}^{mJ+j}\right)\right\| = 0, \tag{93}$$

which then implies that

$$\lim_{m\to\infty}\left\|\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{m})\right\| = 0. \tag{94}$$

There exists a bounded subsequence $\{\mathbf{w}^{m_k}\} \in \{\mathbf{w}^m\}$ which converges to $\mathbf{w}^*$, since $\{\mathbf{w}^m\}$ is bounded. From the argument above, we have

$$E(\mathbf{w}^*) = \lim_{s\to\infty}\widetilde{E}(\mathbf{w}^{m_s}) \tag{95}$$

which yields

$$\widetilde{E}_{\mathbf{w}}(\mathbf{w}^*) = \lim_{s\to\infty}\widetilde{E}_{\mathbf{w}}(\mathbf{w}^{m_s}) = 0. \tag{96}$$

Thus here we get $0 \in \partial\widetilde{E}(\mathbf{w}^*)$. From definition of $\widetilde{E}(\mathbf{w}^m)$, we can find $\partial\widetilde{E}(\mathbf{w}^m) \subseteq \partial E(\mathbf{w}^m)$ by Lemma 4.1. So $0 \in \partial E(\mathbf{w}^*)$. □

**Proof of Theorem 3.2.** From Theorem 3.1, we have

$$0 = \partial\widetilde{E}(\mathbf{w}^*) \subseteq \partial E(\mathbf{w}^*). \tag{97}$$

This means that $\mathbf{w}^*$ is a Clarke stationary point of $E(\mathbf{w}^*)$. If $\{\mathbf{w}^m\}$ has more than two stationary points. Without loss of generality, we can assume there are two different stationary points $\mathbf{w}_1$ and $\mathbf{w}_2$. Without loss of generality, we assume that the first components $w_{11} \neq w_{12}$, $\forall\theta \in (0,1)$, denote $w_\theta = \theta w_{11} + (1-\theta)w_{12}$. Then, there exits a subsequence $\mathbf{w}^{m_j}$ of $\mathbf{w}^m$ such that $\lim\limits_{j\to\infty}\mathbf{w}^{m_j} = \mathbf{w}_\theta$, where $w_\theta$ is the first component of $\mathbf{w}_\theta$. This contradicts the assumption (A3). Thus, $\mathbf{w}^*$ must be the unique Clarke stationary point of $\{\mathbf{w}^m\}$. □

**Proof of Theorem 3.3.** The above Theorem 3.2 has implied that the weight sequence $\{\mathbf{w}^m\}$ converges to a fixed point $\mathbf{w}^*$. By the definitions of (7) and (13), we can find that

$$\begin{aligned}
&\left|E(\mathbf{w}^{mJ+j}) - \widetilde{E}(\mathbf{w}^{mJ+j})\right| \\
&= \Bigg|\lambda \sum_{k=1}^{q} \left(\|\mathbf{u}_{r_k}^{mJ+j}\| - h(\mathbf{u}_{r_k}^{mJ+j}, \alpha_m)\right) \\
&\quad + \lambda \sum_{i=1}^{n} \left(\left\|\mathbf{v}_i^{mJ+j}\right\| - h(\mathbf{v}_i^{mJ+j}, \alpha_m)\right)\Bigg| \\
&\leq \lambda \sum_{k=1}^{q} \left|\|\mathbf{u}_{r_k}^{mJ+j}\| - \frac{\|\mathbf{u}_{r_k}^{mJ+j}\|^2}{2\alpha_m} - \frac{\alpha_m}{2}\right| \\
&\quad + \lambda \sum_{i=1}^{n} \left|\left\|\mathbf{v}_i^{mJ+j}\right\| - \frac{\|\mathbf{v}_i^{mJ+j}\|^2}{2\alpha_m} - \frac{\alpha_m}{2}\right| \qquad (98) \\
&\left(\text{Here } \|\mathbf{u}_{r_k}^{mJ+j}\| \leq \alpha_m \text{ and } \|\mathbf{v}_i^{mJ+j}\| \leq \alpha_m\right) \\
&\leq \lambda \sum_{k=1}^{q} \frac{\left(\|\mathbf{u}_{r_k}^{mJ+j}\| - \alpha_m\right)^2}{2\alpha_m} \\
&\quad + \lambda \sum_{i=1}^{n} \frac{\left(\left\|\mathbf{v}_i^{mJ+j}\right\| - \alpha_m\right)^2}{2\alpha_m} \\
&\leq \lambda q \frac{\alpha_m}{2} + n\lambda \frac{\alpha_m}{2} \to 0. \\
&(\text{since } \alpha_m \to 0 \text{ as } m \to \infty)
\end{aligned}$$

Consequently, we obtain

$$\lim_{m \to \infty} \left|E(\mathbf{w}^m) - \widetilde{E}\left(\mathbf{w}^m\right)\right| = 0. \qquad (99)$$

This completes the proof. □

## 5. Simulations

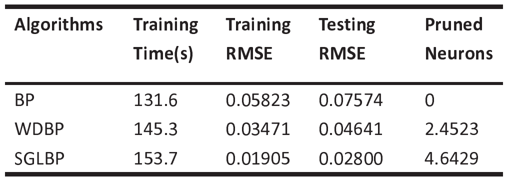
In this section, we provide simulations to compare the performance of the proposed new training algorithm (14)-(21), SGLBP, with the common BP and BP with Weight Decay (WDBP) on two problems: function approximation and nonlinear autoregression. The simulations support the convergence assertion made in Section 3 as well.

## *5.1. Approximation of 'SinC' Function with Noise*

In this example, the three algorithms, BP, WDBP and SGLBP, are used to approximate the "SinC" function, a popular example to demonstrate the performance of intelligent algorithms for regression problems, defined as follows

$$y(x) = \begin{cases} sin(x)/x, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{100}$$

The training set with $2,000$ data ($x^i = 1, 2, \cdots, 2,000$) is randomly generated on the interval $[-10, 10]$ with outputs $y(x^i) + \varepsilon^i$, where $\varepsilon^i$ is the uniform noise distributed in $[-0.04, 0.04]$. The testing set, however, is uniformly created on the interval $[-10, 10]$ with $2,000$ noise-free samples.



| Algorithms | Training Time(s) | Training RMSE | Testing RMSE | Pruned Neurons |
|---|---|---|---|---|
| BP | 131.6 | 0.05823 | 0.07574 | 0 |
| WDBP | 145.3 | 0.03471 | 0.04641 | 2.4523 |
| SGLBP | 153.7 | 0.01905 | 0.02800 | 4.6429 |

Figure 1: Performance comparison for learning of the noisy function SinC.

To compare the three different algorithms, we have designed the identical networks ($2 - 20 - 1$ number of neurons of input, hidden and output layers, separately) with the same initial training parameters to the regression problem. The activation functions of hidden and output layers have been assigned with $tansig(\cdot)$ and $purelin(\cdot)$ functions, respectively. The initial weights (including bias) have been generated with the Nguyen-Widrow algorithm. The learning rate and penalty coefficient are separately set to be $\eta = 0.006$ and $\lambda = 0.009$. The stop criteria for the training is at $40,000$ iterations.

For the sake of comparing the generalization and pruning abilities, the simulations have been repeated 10 times for all the three algorithms. Fig. 1 shows the average results in terms of Training Time, Training RMSE, Testing RMSE and Pruned Neurons, where RMSE means the root mean squared error.

Due to the additional computation burden of the penalty terms, WDBP and SGLBP are more time consuming than the common BP algorithm. However, we observe that the training RMSE of SGLBP is less than those of WDBP and BP, which shows that SGLBP performs a better approximation for the $SinC$ function. The lower testing RMSEs of WDBP and SGLBP demonstrate that they have the stronger generalization ability than BP. From the last column of Fig 1, it shows that SGLBP has the best pruning ability since the group sparse due to the use of Group Lasso penalty.

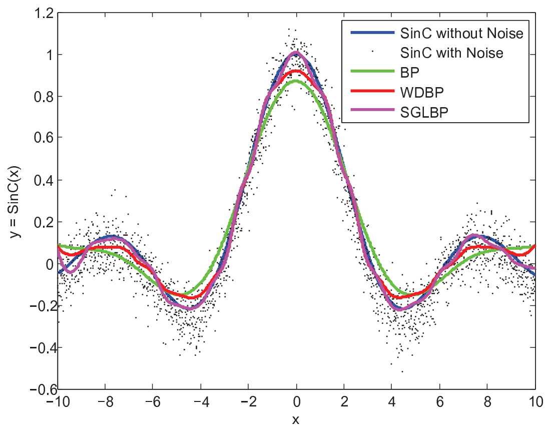


Figure 2: Comparison of the approximation performance for the algorithms: BP, WDBP and SGLBP.

After training, the outputs of the noise-free testing set for the three neural networks have been graphed in Fig. 2. Based on the observation, it is clear that SGBP shows the best approximation performance for $SinC$ function than WDBP and BP.

## *5.2. NAR Problem*

We consider the following two-dimensional nonlinear time series prediction problem [11], defined by

$$\begin{aligned} y(k) = &\big(0.8 - 0.5\exp(-y^2(k-1))\big)\, y(k-1) \\ &- \big(0.3 + 0.9\exp(-y^2(k-1))\big)\, y(k-2) \\ &+ 0.1\sin(\pi y(k-1)), \end{aligned} \tag{101}$$

where $k = 3, 4, \cdots, 1002$.

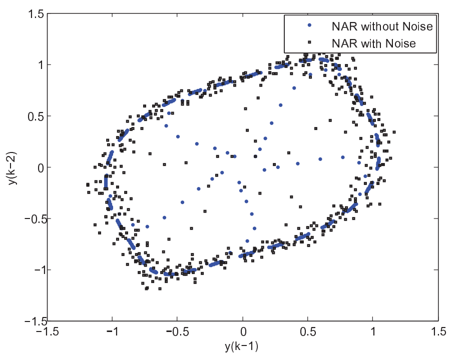


Figure 3: NAR problem: Training data with noise and the original data without noise.

This is a noise-free system which was specified by a limit circle. One thousand samples were generated under the initial condition that $y(1) = y(2) = 0.1$. To verify the stability and noise resistance of the proposed algorithm, the first 500 training samples were added noise $\varepsilon(k)$, where the noise $\varepsilon(k)$ is Gaussian with zero mean and variance 0.05 (cf. Fig. 3). The remaining 500 noise-free samples were used for testing data. According to the previous observations $y(k-1)$ and $y(k-2)$, the proposed algorithm, SGLBP, is trained to predict $y(k)$. In this experiment, the network is designed with three layers

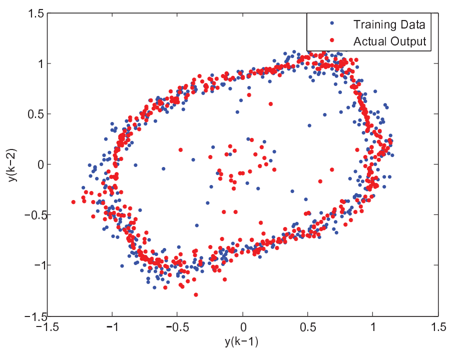


Figure 4: Training data and the corresponding actual output of the trained neural network.

with the architecture $3-13-1$ (number of input, hidden and output neurons which include the bias in the input and hidden layers). Based on the obtained samples, the functions $tansig(\cdot)$ and $purelin(\cdot)$ have been selected as the activation functions of hidden and output layers, respectively. The initial weights have been randomly chosen on the closed interval $[-1.0, 1.0]$ with learning rate $\eta = 0.006$ and the penalty coefficient $\lambda = 0.008$. The stop criteria is that the maximum training iterations reach $40,000$ or the training error is below $0.001$.

In Fig. 4, the blue color points (●) represent the first noisy 500 training samples, while the red color points (●) stand for the corresponding actual outputs of the trained neural network. It shows that the actual outputs are a good approximation for the time series prediction problem (101) except for some of the inner training points.

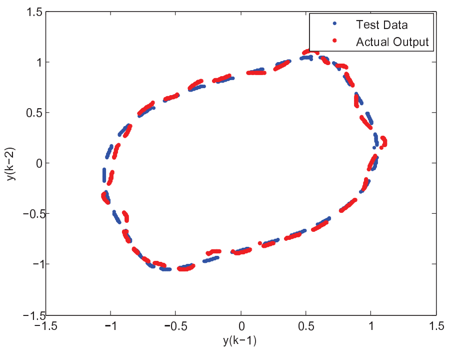


Figure 5: Test data and the corresponding actual output based on the trained neural network.

Generalization ability is one of the important indexes to measure the performance of neural networks. Fig. 5 shows clearly that the proposed algorithm, SGLBP, predicts the testing dataset very well, where (●) and (●) are the ideal test outputs and the predictive outputs, respectively. Fig. 6 demonstrates the convergence behavior of SGLBP.

This simulation shows that SGLBP outperforms the normal BP and WDBP. Furthermore, all these simulations demonstrate the error function of SGLBP decreases to a stable minimum value. Furthermore, the norm of the gradient of error function does approach to zero as iteration increases large enough, which supports the convergence results (cf. Theorem 3.1).

## *5.3. Classification*

In this simulation, the following classification datasets were downloaded from UCI Machine Learning Repository 7, which include 4 binary classification and one multi-class classification problems [28]. Each dataset is ran-

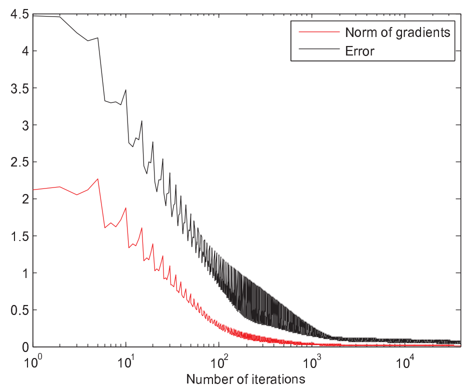
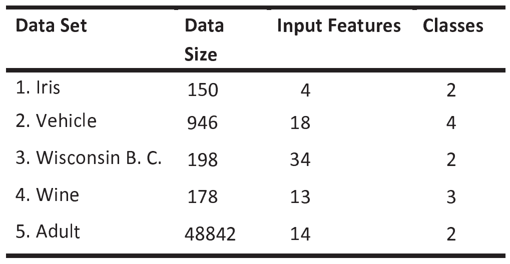


Figure 6: The training error and the norm of gradient of error with respect to weight vector.



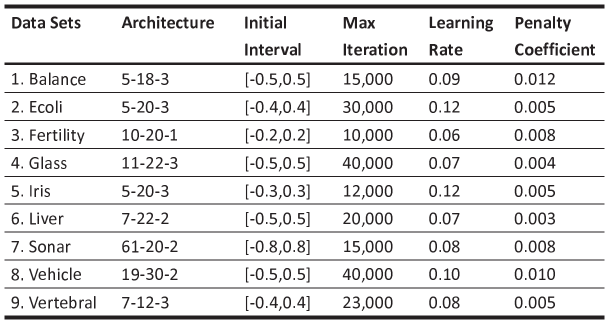
| Data Set | Data Size | Input Features | Classes |
|---|---|---|---|
| 1. Iris | 150 | 4 | 2 |
| 2. Vehicle | 946 | 18 | 4 |
| 3. Wisconsin B. C. | 198 | 34 | 2 |
| 4. Wine | 178 | 13 | 3 |
| 5. Adult | 48842 | 14 | 2 |

Figure 7: Datasets for classification performance comparison.

domly split into training and testing subsets with a fixed percentage, 80% and 20% . For each dataset, we adopt the following normalization technique $x_{new} = \frac{2(x-\bar{x})}{(x_{max}-x_{min})}$ to preprocess the training and testing samples. In terms of the classification datasets, we establish five different FNNs for the two algorithms (Fig. 8). Each of them differs by network architectures, initial weight intervals, number of maximum iterations, learning rates and penalty coefficients, respectively.

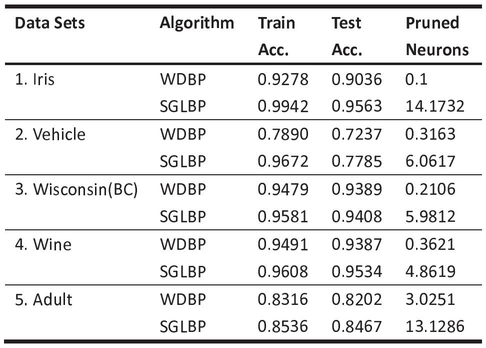
To compare the generalization and pruning abilities of WDBP and the proposed SGLBP, each simulation was repeated 10 times with three fold cross validation process.

For the numerical results, we mainly focus on the three performance metrics which have been listed in the last columns of Fig. 9: the average training accuracy, average testing accuracy and the average number of pruned hidden



| Data Sets | Architecture | Initial Interval | Max Iteration | Learning Rate | Penalty Coefficient |
|---|---|---|---|---|---|
| 1. Balance | 5-18-3 | [-0.5,0.5] | 15,000 | 0.09 | 0.012 |
| 2. Ecoli | 5-20-3 | [-0.4,0.4] | 30,000 | 0.12 | 0.005 |
| 3. Fertility | 10-20-1 | [-0.2,0.2] | 10,000 | 0.06 | 0.008 |
| 4. Glass | 11-22-3 | [-0.5,0.5] | 40,000 | 0.07 | 0.004 |
| 5. Iris | 5-20-3 | [-0.3,0.3] | 12,000 | 0.12 | 0.005 |
| 6. Liver | 7-22-2 | [-0.5,0.5] | 20,000 | 0.07 | 0.003 |
| 7. Sonar | 61-20-2 | [-0.8,0.8] | 15,000 | 0.08 | 0.008 |
| 8. Vehicle | 19-30-2 | [-0.5,0.5] | 40,000 | 0.10 | 0.010 |
| 9. Vertebral | 7-12-3 | [-0.4,0.4] | 23,000 | 0.08 | 0.005 |

Figure 8: Network architecture and the corresponding learning parameters for different datasets.



| Data Sets | Algorithm | Train Acc. | Test Acc. | Pruned Neurons |
|---|---|---|---|---|
| 1. Iris | WDBP | 0.9278 | 0.9036 | 0.1 |
| | SGLBP | 0.9942 | 0.9563 | 14.1732 |
| 2. Vehicle | WDBP | 0.7890 | 0.7237 | 0.3163 |
| | SGLBP | 0.9672 | 0.7785 | 6.0617 |
| 3. Wisconsin(BC) | WDBP | 0.9479 | 0.9389 | 0.2106 |
| | SGLBP | 0.9581 | 0.9408 | 5.9812 |
| 4. Wine | WDBP | 0.9491 | 0.9387 | 0.3621 |
| | SGLBP | 0.9608 | 0.9534 | 4.8619 |
| 5. Adult | WDBP | 0.8316 | 0.8202 | 3.0251 |
| | SGLBP | 0.8536 | 0.8467 | 13.1286 |

Figure 9: Performance comparison for the two pruning algorithms.

neurons. Fig. 9 demonstrates two aspects of the algorithms. The first two performance measures show the network classification capabilities, while the last one, “Pruned Neurons”, evaluates the pruning effect. It is clear to see that the proposed SGLBP algorithm performs much better than the common WDBP algorithm on both classification accuracy and pruning ability.

## 6. Conclusions

We have proposed a new type of penalty functions for neural networks that effectively prunes the neurons at the group level. To overcome the numerical oscillations and nonsmooth problems, smoothing techniques have been applied to approximate the $l_1 - l_2$ norm penalty term. The weak and strong convergence of the suggested smoothing algorithms have been proved which results in the consistent convergence properties for the proposed algorithms with non-differentiable penalty terms. The simulations indicate the usefulness of the new suggested penalization approach.

An important future research topic for incremental FNNs is to analyze the convergence behavior with a constant learning rate. The existing literature on converge analysis shows that the learning rates during training are heavily dependent on the iteration number and decrease to zero as iteration goes on. This then directly leads to reduced efficiency for real experiments. [54] shows an interesting attempt to obtain the convergence results with constant learning rate. Unfortunately, it is only valid for no-hidden layer networks under some complicated constraints. How to guarantee the convergence statements for multilayer networks with constant learning rate and apply for the real applications is one of our next challenging works.

**Acknowledgment**

The authors wish to thank the anonymous reviewers for their insightful comments and suggestions which greatly improved this work.

## References

## References

[1] M. G. Augasta, and T. Kathirvalavakumar, "A novel pruning algorithm for optimizing feedforward neural network of classification problems," *Neural processing letters*, vol. 34, no. 3, pp. 241-258, 2011.

[2] M. G. Augasta, and T. Kathirvalavakumar, "Pruning algorithms of neural networks-a comparative study," *Central European Journal of Computer Science*, vol. 3, no. 3, pp. 105-115, 2013.

[3] L. M. Belue, and K. W. Bauer, "Determining input features for multilayer perceptrons," *Neurocomputing*, vol. 7, no. 2, pp. 111-121, 1995.

[4] D. P. Bertsekas, "Nondifferentiable optimization via approximation," *Mathematical Programming Studiy*, vol. 3 pp. 1-25, 1975.

[5] W. Bian, and X. Xue, "Subgradient-based neural networks for nonsmooth nonconvex optimization problems," *Neural Networks, IEEE Transactions on*, vol. 20, no. 6, pp. 1024-1038, 2009.

[6] W. Bian, and X. Chen, “Worst-Case Complexity of Smoothing Quadratic Regularization Methods for Non-Lipschitzian Optimization,” *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1718-1741, 2013.

[7] W. Bian, and X. Chen, “Smoothing neural network for constrained non-Lipschitz optimization with applications,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 3, pp. 399-411, 2012.

[8] W. Bian, and X. Chen, “Neural Network for Nonsmooth, Nonconvex Constrained Minimization Via Smooth Approximation,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 545-556, 2014.

[9] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.

[10] D. Chakraborty, and N. R. Pal, “A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning,” *Neural Networks, IEEE Transactions on*, vol. 14, no. 1, pp. 1-14, 2003.

[11] S. Chen, “Local regularization assisted orthogonal least squares regression,” *Neurocomputing*, vol. 69, nos. 4-6, pp. 559-585, 2006.

[12] X. Chen, “Smoothing methods for nonsmooth, nonconvex minimization,” *Mathematical Programming*, vol. 134, no. 1, pp. 71-99, 2012.

[13] X. Chen, F. Xu, and Y. Ye, “Lower Bound Theory of Nonzero Entries in Solutions of $\ell_2$-$\ell_p$ Minimization,” *SIAM Journal on Scientific Computing*, vol. 32, no. 5, pp. 2832-2852, 2010.

[14] X. Chen, and W. Zhou, “Smoothing Nonlinear Conjugate Gradient Method for Image Restoration Using Nonsmooth Nonconvex Minimization,” *SIAM Journal on Imaging Sciences*, vol. 3, no. 4, pp. 765-790, 2010.

[15] Z. Chen, and S. Haykin, “On Different Facets of Regularization Theory,” *Neural Comput.,* vol. 14, no. 12, pp. 2791-2846, 2002.

[16] F. H. Clarke, *Optimization and Nonsmooth Analysis.* New York: Wiley, 1983

[17] A. P. Engelbrecht, “A new pruning heuristic based on variance analysis of sensitivity information,” *Neural Networks, IEEE Transactions on*, vol. 12, no. 6, pp. 1386-1399, 2001.

[18] L. Fletcher, V. Katkovnik, F. E. Steffens, and A. P. Engelbrecht, “Optimizing the number of hidden nodes of a feedforward artificial neural network,” in *Proc. IEEE world congress on computational intelligence, The international joint conference on neural networks*, 1998, pp. 1608-1612

[19] J. Friedman, T. Hastie, and R. Tibshirani, “A note on the group lasso and a sparse group lasso,” *Preprint arXiv:1001.0736*, 2010.

[20] M. Forti, P. Nistri, and M. Quincampoix, “Generalized neural network for nonsmooth nonlinear programming problems,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 9, pp. 1741-1754, 2004.

[21] M. Hagiwara, “A simple and effective method for removal of hidden units and weights,” *Neurocomputing*, vol. 6, no. 2, pp. 207-218, 1994.

[22] S. J. Hanson, and L. Y. Pratt, ”Comparing biases for minimal network construction with back-propagation,” in *Advances in Neural Information Processing Systems,* 1989, 177-185.

[23] S. S. Haykin, *Neural networks : a comprehensive foundation,* 2nd, Englewood Cliffs, NJ, USA: Prentice Hall, 1999.

[24] S. Haykin, *Neural networks and learning machines.* McMaster University. Hamilton, Ontario, Canada.: Prentice Hall, 2009.

[25] T. Heskes, and W. Wiegerinck, “A theoretical comparison of batch-mode, on-line, cyclic, and almost-cyclic learning,” *Neural Networks, IEEE Transactions on*, vol. 7, no. 4, pp. 919-925, 1996.

[26] G. E. Hinton, “Connectionist learning procedures,” *Artificial intelligence*, vol. 40, no. 1, pp. 185-234, 1989.

[27] W. W. Hsieh, *Machine Learining Methods in the Envriomental Sciences.* Cambridge, U.K.: Cambridge Univ. Press, 2009

[28] http://archive.ics.uci.edu/ml/

[29] S. C. Huang, and Y. F. Huang, "Bounds on the number of hidden neurons in multilayer perceptrons," *Neural Networks, IEEE Transactions on*, vol. 2, no. 1, pp. 47-55, 1991.

[30] T. Q. Huynh, and R. Setiono, "Effective neural network pruning using cross-validation," in *Proc. IEEE international joint conference on neural networks (IJCNN)*,2005, pp. 972-977.

[31] M. Z. Iskandarani, "A novel Approach to System Security using Derived Odor Keys with Weight Elimination Neural Algorithm (DOK-WENA)," *Trans. Mach. Learn. Artif. Intell.,* vol. 2, no. 2, pp. 20-31, 2014.

[32] Z. Li, W. Wu, and Y. Tian, "Convergence of an online gradient method for feedforward neural networks with stochastic inputs," *Journal of Computational and Applied Mathematics*, vol. 163, no. 1, pp. 165-176, 2004.

[33] W. Lu, and J. Wang, "Convergence analysis of a class of nonsmooth gradient systems," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 55, no. 11, pp. 3514-3527, 2008.

[34] P. May, E. Zhou, and C. Lee, "A comprehensive evaluation of weight growth and weight elimination methods using the tangent plane algorithm," *Int. J. Adv. Comput. Sci. & Appl.,*, vol. 4, no. 6, pp. 149-56, 2013.

[35] J. O. Moody, and P. J. Antsaklis, "The dependence identification neural network construction algorithm," *Neural Networks, IEEE Transactions on*, vol. 7, no. 1, pp. 3-15, 1996.

[36] J. E. Moody and Thorsteinn S. Rögnvaldsson, "Smoothing regularizers for projective basis function networks," CSETech. 1996.

[37] B. D. Ripley, *Pattern Recognition and Neural Network*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[38] D. Sabo, and X.-H. Yu, "Neural network Dimension Selection for dynamical system identification," in *Proc. IEEE international conference on control applications*, 2008, pp. 972-977.

[39] R. Setiono, and L. C. K. Hui, “Use of a quasi-Newton method in a feedforward neural network construction algorithm,” *Neural Networks, IEEE Transactions on*, vol. 6, no. 1, pp. 273-277, 1995.

[40] R. Setiono, “A penalty-function approach for pruning feedforward neural networks,” *Neural computation*, vol. 9, no. 1, pp. 185-204, 1997.

[41] J. Sum, C. S. Leung, and K. Ho, “Convergence analyses on on-line weight noise injection-based training algorithms for MLPs,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 11, pp. 1827-1840, 2012.

[42] W. Sun, and Y.-X. Yuan, *Optimization theory and methods: nonlinear programming.* Springer Science & Business Media, 2006.

[43] V. Tadic, and S. Stankovic, ”Learning in neural networks by normalized stochastic gradient algorithm: local convergence,” in *Proc. Seminar neural networks application electronic engineering,* 2000, pp. 11-17.

[44] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267-288, 1996.

[45] J. Wang, W. Wu, and J. M. Zurada, ”Boundedness and convergence of MPN for cyclic and almost cyclic learning with penalty,” in *Proc. IEEE International Joint Conference on Neural Networks (IJCNN)*, 2011, pp. 125-132.

[46] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, “Generalization by weight-elimination applied to currency exchange rate prediction,” in *Proc. IEEE International Joint Conf. Neural Netw. (IJCNN)*, 1991, pp. 837-841.

[47] D. Whitley, T. Starkweather, and C. Bogart, “Genetic algorithms and neural networks: Optimizing connections and connectivity,” *Parallel computing*, vol. 14, no. 3, pp. 347-361, 1990.

[48] W. Wu, J. Wang, M. Cheng, and Z. Li, “Convergence analysis of online gradient method for BP neural networks,” *Neural Networks*, vol. 24, no. 1, pp. 91-98, 2011.

[49] W. Wu, and Y. Xu, "Deterministic convergence of an online gradient method for neural networks," *Journal of Computational and Applied Mathematics*, vol. 144, no. 1, pp. 335-347, 2002.

[50] L. Wu, and J. Moody, "A smoothing regularizer for feedforward and recurrent neural networks," *Neural Computation*, vol. 8, no. 3, pp. 461-489, 1996.

[51] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," Proceedings of COMPSTAT'2010. Physica-Verlag HD, pp. 177-186, 2010.

[52] W. Xu, "Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent," *Computer Science*, 2011.

[53] S. Shalev-Shwartz, "Online Learning and Online Convex Optimization," *Foundations & Trends in Machine Learning*, vol. 4, no.2, pp. 107-194, 2012.

[54] L. Xu, J. Chen, D. Huang, J. Lu, and L. Fang, "Analysis of boundedness and convergence of online gradient method for two-layer feedforward neural networks," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 8, pp. 1327-1337, 2013.

[55] Z. B. Xu, R. Zhang, and W.-F. Jing, "When does online BP training converge?," *Neural Networks, IEEE Transactions on*, vol. 20, no. 10, pp. 1529-1539, 2009.

[56] H. Zhang, W. Wu, F. Liu, and M. Yao, "Boundedness and convergence of online gradient method with penalty for feedforward neural networks," *Neural Networks, IEEE Transactions on*, vol. 20, no. 6, pp. 1050-1054, 2009.

[57] M. Yuan, and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49-67, 2006.

[58] J. Zhang, and A. J. Morris, "A sequential learning approach for single hidden layer neural networks," *Neural networks*, vol. 11, no. 1, pp. 65-80, 1998.

[59] H. Zou, “The adaptive lasso and its oracle properties,” *Journal of the American statistical association*, vol. 101, no. 476, pp. 1418-1429, 2006.