A Batch Variable Learning Rate Gradient Descent Algorithm With the Smoothing $L_{1/2}^1$ Regularization for Takagi-Sugeno Models

YUNLONG LU1, WENYU LI1, AND HONGWEI WANG2

1School of Mathematics and Statistics, Beihua University, Jilin 132013, China
2Department of Mathematics and Physics, Texas A&M International University, Laredo, TX 78041, USA

Corresponding author: Wenyu Li (liwenyu82@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61473059, in part by the doctoral research project establishment and start-up fund of Beihua University, and in part by the Science and Technology Development Plan Project of Jilin Province under Grant 20180519011JH and Grant 20190303132SF.

ABSTRACT A batch variable learning rate gradient descent algorithm is proposed to efficiently train a neuro-fuzzy network of zero-order Takagi-Sugeno inference systems. By using the advantages of regularization, the smoothing $L_{1/2}$ regularization is utilized to find more appropriate sparse network. Combining the second-order information of the smoothing error function, a variable learning rate is chosen along the steep descent direction, which avoids line search procedure and may reduce the cost of computation. In order to appropriately adjust the Lipschitz constant of the smoothing error function in the learning rate, a new scheme is proposed by introducing a hyper-parameter. Also the article applies the modified secant equation for estimating the Lipschitz constant, which makes the algorithm greatly reduce the oscillating phenomenon and improve the robustness. Under appropriate assumptions, a convergent result of the proposed algorithm is also given. Simulation results for two identification and classification problems show that the proposed algorithm has better numerical performance and promotes the sparsity capability of the network, compared with the common batch gradient descent algorithm and a variable learning rate gradient-based algorithm.

INDEX TERMS Zero-order Takagi-Sugeno models, variable learning rate, gradient descent method, smoothing $L_{1/2}$ regularization, modified secant equation.

I. INTRODUCTION

Regularization is one of the important approaches for the overfitting problem. A regularization term is generally introduced on the basis of minimizing the empirical risk to limit the model capabilities so that they do not excessively minimize empirical risk. Regularization has been applied successfully in neural networks for the sparsity of weights and hidden layer units [1]–[5]. The $L_p$ regularization is commonly used, leading to the following minimizing problem

$$
\min_w \tilde{E}(w) + \lambda \|w\|_p^p,
$$

where $\tilde{E}(w)$ is an error function depending on the weights $w$ of networks, $\|w\|_p = \left(\sum_{i=1}^{N} |w_i|^p\right)^{1/p}$, and $\lambda$ is the regularization parameter, which is used to control the strength of regularization. We know that better network sparsity often obtains better generalization performance. The $L_{1/2}$ regularization [6] is a typical regularization, which is capable of obtaining even sparser solution and achieves better generalization performance than the $L_2$ and $L_1$ regularizations. However, the $L_{1/2}$ regularization is nonconvex, nonsmooth, and non-Lipschitz. Thus, the gradient-based learning algorithms can not be used directly to train neural networks. By introducing smoothing technique, Wu et al. [1], [2] have proposed the standard gradient descent algorithms with the smoothing $L_{1/2}$ regularization for training BP feedforward neural network. It is shown that a better pruning of the network is obtained than the $L_1$ and $L_2$ regularizations. In particular, a good review about the study of the smoothing $L_{1/2}$ regularization in different type of networks, such as BP feedforward neural network, high order neural network, parallel feedforward neural network and Takagi-Sugeno (T-S) fuzzy systems, can be found in literature [4]. Neuro-fuzzy systems
are important intelligent systems, which combine artificial neural networks and fuzzy systems, and have been applied successfully in various fields such as pattern recognition, image processing and feature extraction, soft sensors, control system and system identification [7]–[10]. Many common batch gradient descent algorithms are proposed to train neuro-fuzzy networks for neuro-fuzzy systems [11]–[16]. For training the network, it needs to minimize a high nonlinear and ill-condition problem, the common batch gradient descent algorithm is simple, however, it usually converges slowly and exhibits oscillating phenomenon. These drawbacks have been discussed in literature [17]–[19]. To address the drawbacks, some gradient-based algorithms have been developed. One class is to use smoothing techniques to overcome serious numerical oscillations. Recently, by using the smoothing $L_{1/2}$ regularization, a constant learning rate gradient descent smoothing algorithm [3] is proposed to reduce the adverse effects of high non-linearity of the error function for T-S fuzzy models, in which the learning rate is usually chosen by a smaller value. Another class of typical algorithms is to use the second-order information of the error function to overcome the adverse effects of high non-linearity and ill-condition. By combining a modified secant equation to generate the searching direction, it approximates the second order curvature of the error function with a high-order accuracy, the efficient conjugate gradient algorithms have been proposed to train BP networks [5], [20]. In optimization community, by estimating the part information of the second-order Hessian matrix of the error function to compute the learning rate without line search, the variants of the common gradient descent algorithm [21], [22] have been proposed, but figuring out how to ingeniously adjust the estimation of the Lipschitz constant or choose suitable positive definite matrices is not easy.

Therefore, for neuro-fuzzy systems, in order to reduce the adverse effects of high non-linearity and ill-condition of the error function and improve the numerical performance of the algorithm and generalization capacity of the network, we shall use the second-order information from the iterations to define a learning rate without line search, and further propose a batch gradient-based algorithm with the smoothing $L_{1/2}$ regularization. The advantages of the proposed algorithm and the main contribution of the paper are as follows:

1. We have proposed a new scheme for adjusting the Lipschitz constant to update the learning rate by introducing a hyper-parameter. And the modified secant equation [25], which can lead to a good property that the hereditary positive definiteness of the approximation of the Hessian matrix is always guaranteed, is also used for estimating the Lipschitz constant. Then the algorithm with a modified secant equation makes better use of second-order information than the algorithms with a common secant equation, which makes the proposed algorithm greatly reduce the oscillating phenomenon, and enhance the robustness.

2. Also compared with the common batch gradient descent neuro-fuzzy learning algorithm [3], the proposed algorithm can obtain lower error, effectively reduce the oscillation phenomenon and find the appropriate sparse network. Thus, by appropriately applying the second-order information and the smoothing $L_{1/2}$ regularization, the proposed algorithm shows better sparsity and generalization capacity.

The numerical simulations have demonstrated the performance of the algorithm. Moreover, under the assumptions, the global convergent result is given.

The rest of the paper is organized as follows. In Section II, we describe the zero-order T-S fuzzy system. In Section III, we propose a variable learning rate gradient algorithm with the smoothing $L_{1/2}$ regularization. In Section IV, a convergent result is given. In Section V, simulation results are presented. Some conclusions are summarized in Section VI.

II. ZERO-ORDER TAKAGI-SUGENO INFERENCE SYSTEM

In this section, we use zero-order T-S inference system based on fuzzy neural network. Its topological structure of the zero-order T-S neuro-fuzzy network is shown in Figure 1. We consider a four-layer network, which includes the input layer, linguistic variable layer, fuzzy rule layer and output layer with an output node. The case with more output nodes can also be proposed easily by a similar way, which we do not mention here.

The zero-order T-S inference system [23], [24] is a common fuzzy system. Its fuzzy rule base is composed of a series of IF-THEN fuzzy rules. The $i$-th rule, denoted as $R_i$, $i = 1, 2, \ldots, n$, is represented in the following form:

$$R_i : \text{IF} \ x_1 \ \text{is} \ A_{1i} \ \text{and} \ x_m \ \text{is} \ A_{mi} \ \text{THEN} \ z \ \text{is} \ y_i;$$

where $x_1, x_2, \ldots, x_m$ are the inputs to the system, $A_{ij}$ denotes a fuzzy set of $x_i$, $i = 1, 2, \ldots, m$, $z$ is the system output variable, and the value $y_i$ is a real number. Here we select the

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{Topological structure of the zero-order T-S inference system.}
\end{figure}
TABLE 1. Numerical results for identifying $y = \sin(\pi x)$.

| Method       | $\eta$ | $\lambda$ | Average error of training patterns/Std | Average error of test patterns/Std | ACAN | Average of training time (s) |
|--------------|--------|-----------|---------------------------------------|-----------------------------------|------|-----------------------------|
| GSL$_{1/2}$ | 0.001  | 0.03      | 5.86e-04/6.08e-04                     | 7.90e-04/7.72e-04                 | 3.2  | 0.59                        |
| Algorithm A3 | 0.01   | 0.003/8.90e-04 | 0.003/8.00e-011                     | 1.5                               | 0.49 |
| VLRGSL$_{1/2}$ | 0.001 | 0.03      | 4.42e-04/3.27e-04                     | 5.94e-04/3.75e-04                 | 7.6  | 0.51                        |
| MSE-VLRGSL$_{1/2}$ | 0.001 | 0.03      | 4.34e-04/3.07e-04                     | 5.85e-04/3.51e-04                 | 7.8  | 0.65                        |
| GSL$_{1/2}$ | 0.002  | 0.03      | 3.73e-04/1.08e-04                     | 5.36e-04/7.36e-04                 | 3.8  | 0.41                        |
| Algorithm A3 | 0.002 | 0.03      | 3.62e-04/1.1e-04                      | 5.14e-04/7.39e-04                 | 3.9  | 0.47                        |
| VLRGSL$_{1/2}$ | 0.002 | 0.03      | 3.75e-04/3.07e-05                     | 5.29e-04/9.93e-05                 | 7.8  | 0.49                        |
| MSE-VLRGSL$_{1/2}$ | 0.002 | 0.03      | 3.71e-04/7.15e-05                     | 5.25e-04/9.97e-05                 | 8.5  | 0.62                        |
| GSL$_{1/2}$ | 0.003  | 0.03      | 7.32e-04/3.80e-04                     | 0.001/9.54e-04                    | 3.2  | 0.61                        |
| Algorithm A3 | 0.003 | 0.03      | 3.77e-04/1.84e-04                     | 5.32e-04/1.28e-04                 | 3.2  | 0.50                        |
| VLRGSL$_{1/2}$ | 0.003 | 0.03      | 3.62e-04/1.05e-04                     | 5.10e-04/3.27e-04                 | 7.8  | 0.51                        |
| MSE-VLRGSL$_{1/2}$ | 0.003 | 0.03      | 3.62e-04/5.34e-05                     | 5.11e-04/7.33e-05                 | 8.4  | 0.61                        |
| GSL$_{1/2}$ | 0.004  | 0.03      | 0.001/8.00e-04                        | 0.001/8.00e-04                    | 1    | 0.42                        |
| Algorithm A3 | 0.004 | 0.03      | 3.58e-04/4.35e-04                     | 5.04e-04/4.90e-04                 | 5.0  | 0.51                        |
| VLRGSL$_{1/2}$ | 0.004 | 0.03      | 3.43e-04/1.04e-04                     | 4.82e-04/1.46e-04                 | 6.5  | 0.54                        |
| MSE-VLRGSL$_{1/2}$ | 0.004 | 0.03      | 3.38e-04/9.66e-05                     | 4.76e-04/1.35e-04                 | 7.8  | 0.63                        |
| GSL$_{1/2}$ | 0.005  | 0.03      | 0.202/0.001                          | 0.197/0.0976                      | 1.3  | 0.44                        |
| Algorithm A3 | 0.005 | 0.03      | 3.43e-04/3.67e-04                     | 5.44e-04/4.27e-04                 | 4.1  | 0.51                        |
| VLRGSL$_{1/2}$ | 0.005 | 0.03      | 3.45e-04/2.39e-04                     | 4.90e-04/3.53e-04                 | 4.7  | 0.51                        |
| MSE-VLRGSL$_{1/2}$ | 0.005 | 0.03      | 3.30e-04/2.17e-04                     | 4.72e-04/2.36e-04                 | 4.9  | 0.61                        |
| GSL$_{1/2}$ | 0.001  | 0.01      | 3.33e-04/8.32e-04                     | 4.39e-04/4.64e-04                 | 1.5  | 0.39                        |
| Algorithm A3 | 0.001 | 0.01      | 0.001/8.36e-04                       | 0.0023/0.0010                     | 0.6  | 0.45                        |
| VLRGSL$_{1/2}$ | 0.001 | 0.01      | 1.49e-04/8.65e-04                     | 2.06e-04/8.25e-04                 | 3.4  | 0.48                        |
| MSE-VLRGSL$_{1/2}$ | 0.001 | 0.01      | 1.47e-04/5.47e-04                     | 2.03e-04/7.56e-04                 | 3.5  | 0.57                        |
| GSL$_{1/2}$ | 0.001  | 0.009     | 3.21e-04/6.35e-04                     | 4.23e-04/4.46e-04                 | 1.4  | 0.39                        |
| Algorithm A3 | 0.001 | 0.009     | 0.001/8.27e-04                       | 0.0023/9.89e-04                  | 0.6  | 0.45                        |
| VLRGSL$_{1/2}$ | 0.001 | 0.009     | 1.39e-04/6.75e-05                     | 1.92e-04/9.01e-05                 | 3.3  | 0.46                        |
| MSE-VLRGSL$_{1/2}$ | 0.001 | 0.009     | 1.38e-04/4.30e-05                     | 1.89e-04/4.66e-05                 | 3.5  | 0.57                        |
| GSL$_{1/2}$ | 0.001  | 0.008     | 3.09e-04/4.39e-04                     | 4.08e-04/7.26e-04                 | 1.3  | 0.37                        |
| Algorithm A3 | 0.001 | 0.008     | 0.0017/1.81e-04                      | 0.0021/9.89e-04                  | 0.6  | 0.45                        |
| VLRGSL$_{1/2}$ | 0.001 | 0.008     | 1.29e-04/4.05e-05                     | 1.78e-04/4.66e-05                 | 2.9  | 0.46                        |
| MSE-VLRGSL$_{1/2}$ | 0.001 | 0.008     | 1.28e-04/3.17e-05                     | 1.75e-04/4.33e-05                 | 3.2  | 0.57                        |

Next, combining IF-THEN fuzzy rules (2) and the Gauss functions (3), we clarify the meaning of each layer in Figure 1, as follows:

Layer 1: Each node in this layer represents one input variable of an observation data $x = (x_1, x_2, \cdots, x_m)^T$.

Layer 2: In this layer, each node represents the membership function of a linguistic variable, and the Gauss membership function (3) is to calculate the value of the membership function of each input component belonging to the fuzzy set of linguistic variables. The connecting weights from Layer 1 and Layer 2 are regarded as the centers and the widths of the Gauss membership function, respectively.

Layer 3: In this layer of the network, each node represents a fuzzy rule, and its main function is to match the antecedent part of the fuzzy rule, and the agreement of the $i$-th antecedent part is usually computed by using the product T-norm, as follows:

$$h_i = \prod_{l=1}^{n} A_{li}(x_l),$$

where $A_{li}$ represents the center of the Gauss function $A_{li}(x_l)$.

Layer 4: The output layer has an output node, which is a linear combination of the results from layer 3, and the output $z$ is computed as follows:

$$z = \sum_{i=1}^{n} h_i y_i,$$

where $y_i, i = 1, 2, \cdots, n$, are the connecting weights between Layer 3 and Layer 4.
performance. Next, we will propose a variable learning rate gradient algorithm.

For the sake of convenience, the Hadamard product is used to describe the following contents, which is the component-wise product of matrices, denoted by “·” here.

Suppose that the number of input nodes is $m$. The set of training patterns $\{x_j, O_j\}_{j=1}^J \subset \mathbb{R}^m \times \mathbb{R}$ is supplied to the network, where $O_j$ is the desired output for the $j$-th training pattern $x_j$ and $J$ is the number of training patterns. The fuzzy rule base is provided by (2). Let

$w = (a_0^T, a_1^T, \ldots, a_n^T, b_1^T, \ldots, b_n^T)^T \in \mathbb{R}^{n(1+2m)}$

be all weight vectors relevant to the network except the constant weights, where the vector $a_0 = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$ is the weight vector connecting Layer 3 and Layer 4, and vectors

$a_i = (a_{i1}, a_{i2}, \ldots, a_{im})^T \in \mathbb{R}^m,$

$b_i = (b_{i1}, b_{i2}, \ldots, b_{im})^T \in \mathbb{R}^m,$

$i = 1, 2, \ldots, n,$

are the centers and the reciprocals of the widths of the corresponding Gaussian membership functions, respectively.

For simplicity, the following vector-valued function is introduced by

$h : \mathbb{R}^m \rightarrow \mathbb{R}^n,$

$h(x) = (h_1(x), h_2(x), \ldots, h_n(x))^T,$

for $x = (x_1, x_2, \ldots, x_m)^T \in \mathbb{R}^m$. Hence, the error function $E(w)$ in (1) is given by

$\tilde{E}(w) = \frac{1}{2} \sum_{j=1}^J (O^j - \tilde{z}^j)^2$

$= \sum_{j=1}^J g_j(a_0 \cdot h^j), \quad (6)$

where $\tilde{z}^j$ is the fuzzy reasoning result for the $j$-th training pattern $x^j$ and

$h^j \triangleq h(x^j) = (h_1^j, h_2^j, \ldots, h_n^j), \quad h_i^j \triangleq h_i(x^j), \quad i = 1, 2, \ldots, n,$

$g_j(t) = \frac{1}{2} (O^j - t)^2, \quad j = 1, 2, \ldots, J, \quad t \in \mathbb{R}. \quad (7)$
The gradient of the error function $\vec{E}(w)$ with respect to $a_0, a_i, b_i, i = 1, 2, \cdots, n$, are given respectively by

$$
\vec{E}(w)_{a_0} = \sum_{j=1}^{J} g_j(a_0 \cdot h')h',
$$

$$
\vec{E}(w)_{a_i} = 2 \sum_{j=1}^{J} g_j'(a_0 \cdot h')y_jh'_i((x^j - a_i) \cdot b_i),
$$

$$
\vec{E}(w)_{b_i} = -2 \sum_{j=1}^{J} g_j'(a_0 \cdot h')y_jh'_i((x^j - a_i) \cdot b_i),
$$

(8)

Therefore, the gradient of the error function $\vec{E}(w)$ with respect to $w$ is given by

$$
\nabla \vec{E}(w) = (\vec{E}_{a_0}(w), \vec{E}_{a_1}(w), \cdots, \vec{E}_{a_n}(w), \cdots, \vec{E}_{b_0}(w), \cdots, \vec{E}_{b_i}(w), \cdots)^T.
$$

Consider a polynomial smoothing function of $|x|$, the approximating function is given by

$$
f(x) = \begin{cases} 
-x, & \text{if } x \leq -v, \\
\frac{1}{8v^3}x^4 + \frac{3}{4v^2}x^2 + \frac{3}{8}, & \text{if } -v < x < v, \\
x, & \text{if } x \geq v,
\end{cases}
$$

(9)

where $v$ is a small positive constant. The smoothing function has the following properties:

$$
f(x) \in \left[\frac{3}{8v}, +\infty\right), f'(x) \in [-1, 1], f''(x) \in [0, \frac{3}{2v^3}].
$$

(10)

Using the above smoothing approximation (9), we can obtain the smoothing approximation of the $L_{1/2}$ regularization $\|w\|_{1/2}^{1/2}$ so that the original error function in (1) is approximated by the following function

$$
E(w) = \vec{E}(w) + \lambda F(w),
$$

(11)

where $\lambda$ is a regularization parameter, and

$$
F(w) = \sum_{i=1}^{n} f(y_i)^2 + \sum_{i=1}^{n} \sum_{k=1}^{m} f(a_{ik})^2 + \sum_{i=1}^{n} \sum_{k=1}^{m} f(b_{ik})^2.
$$

Correspondingly, the gradient of the smoothing approximating function $\vec{E}(w)$ on $w$ is denoted by

$$
\nabla \vec{E}(w) = \nabla \vec{E}(w) + \lambda \nabla F(w),
$$

(12)

where

$$
\nabla F(w) = \begin{pmatrix} f'(y_1) & f'(y_n) & f'(a_{11}) & \cdots & f'(a_{1m}) \\
2f'(y_1)^2 & 2f'(y_n)^2 & 2f'(a_{11})^2 & \cdots & 2f'(a_{1m})^2 \\
2f'(a_{11}) & 2f'(a_{1m}) & f'(a_{21}) & \cdots & f'(a_{2m}) \\
2f'(b_{11}) & 2f'(b_{1m}) & 2f'(b_{21}) & \cdots & 2f'(b_{2m}) \\
2f'(b_{11})^2 & 2f'(b_{1m})^2 & 2f'(b_{21})^2 & \cdots & 2f'(b_{2m})^2 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
2f'(b_{11})^m & 2f'(b_{1m})^m & 2f'(b_{21})^m & \cdots & 2f'(b_{2m})^m \\
\end{pmatrix}.
$$

(13)

From [22], we know that figuring out how to adjust the estimation of the Lipschitz constant of $\nabla \vec{E}(w)$ is not easy. Next, a hyper-parameter $c$ is introduced to appropriately adjust the estimate value of the Lipschitz constant in proposed algorithm, and we will firstly propose the variable learning rate gradient algorithm with smoothing $L_{1/2}$ regularization, denoted as VLRGSL$_{1/2}$. For convenience, we denote $E_k \triangleq E(w_k)$ and $\nabla E_k \triangleq \nabla E(w_k)$.
Algorithm 1: VLRGSL_{1/2}

Step 1. Randomly choose the initial weights \( w_0 \in \mathbb{R}^{n(1+2m)} \), and the terminal number of epochs \( T_{\text{ol}} > 0 \), the maximum number of epochs \( K_{\max} \), the constants \( \eta, L_0, c \in (0, 1) \), the regularization parameter \( \kappa \). Set \( k = 0 \).

Step 2. Compute the error function value \( E_k \) in (11) and its gradient \( \nabla E_k \) in (12).

Step 3. If \( \|\nabla E_k\| < T_{\text{ol}} \) or \( k > K_{\max} \), then stop. Otherwise, if \( k \geq 1 \), compute \( L_k \) by the following formula:

\[
L_k = c \max \left\{ L_{k-1}, \frac{\|\xi_{k-1}\|}{\|s_{k-1}\|} \right\},
\]

where \( \xi_{k-1} = \nabla E(w_{k-1}) - \nabla E(w_{k-1}) \) and \( s_{k-1} = w_k - w_{k-1} \).

Step 4. \( w_{k+1} = \frac{\omega_k}{L_k} \nabla E_k \), where \( \omega_k = \frac{n}{L_k^2} \).

Step 5. Set \( k = k + 1 \), and go to Step 2.

In order to adjust \( L_k \), we introduce the hyper-parameter \( c \) in (14). It can be observed that \( L_k \) is not \( c \) times of \( L_0^2 \), where \( L_0^2 \) denotes the estimation of the Lipschitz constant in [22]. For example, when \( k = 2 \), we have

\[
L_2 = c \max \{ c \max \{ L_0, \frac{\|\xi_0\|}{\|s_0\|}, \frac{\|\xi_1\|}{\|s_1\|} \} \}.
\]

It is easy to see that the above \( L_2 \) is not equal to \( cL_0^2 \) when \( c \in (0, 1) \) except for when \( c = 1 \). The new adjustment scheme has been shown to be effective in numerical experiments.

Using the second-order information, based on the modified secant equations, the conjugate gradient algorithms [5], [20] have been confirmed to improve numerical performance for training the neural networks. Like the modified Newton method’s idea, a modified BFGS method [25] is proposed with a modified secant equation, and the equation is given as follows:

\[
B_{k+1} s_k = \tilde{\xi}_k = \xi_k + t_k s_k,
\]

where the matrix \( B_k \) is an approximation of the Hessian of the smoothing error function \( E(w) \) at \( w_k \), \( \xi_k = \nabla E(w_{k+1}) - \nabla E(w_k), s_k = w_{k+1} - w_k \) and

\[
t_k = r \|\nabla E(w_k)\|^\mu + \max\left\{ -\frac{s_k^T \xi_k}{\|s_k\|^2}, 0 \right\}
\]

with \( r > 0 \) and \( \mu > 0 \). It is easy to know that

\[
s_k^T \xi_k \geq \|\nabla E(w_k)\| \|s_k\|^2 > 0,
\]

which means that the hereditary positive definiteness of \( B_k \) is always guaranteed. Thus, in order to make better use of the second-order information, combining the modified secant equations (MSE) (16), \( L_k \) is modified as follows:

\[
L_k = c \max \left\{ L_{k-1}, \frac{\|\xi_{k-1}\|}{\|s_{k-1}\|} \right\},
\]

the corresponding algorithm is denoted as MSE+VLRGSL_{1/2}.

**IV. CONVERGENT RESULTS**

In the section, we present a convergent result of Algorithm 1. The result is easily concluded by Theorem 3.1 in [22] and we omit its proof here. Some assumptions are needed for the convergent result.

(A1) The level set \( S = \{ E(w) \leq E(w_0) \} \) is bounded.

(A2) The gradient \( \nabla E(w) \) is Lipschitz continuous in a neighborhood \( N \) of the level set \( S \), i.e. there exists a constant \( L \geq 0 \) such that

\[
\|\nabla E(w_1) - \nabla E(w_2)\| \leq L \|E(w_1) - E(w_2)\|,
\]

for any \( w_1, w_2 \in N \).

**Theorem 1:** Suppose that Assumptions (A1) and (A2) hold, the infinite sequence \{\( w_k \)\} is generated by Algorithm 1, and there exists a constant \( \zeta \) such that \( \zeta \in (\frac{x}{2}, 1) \), \( L_k \geq \zeta L \), and \( \sum_{k=0}^{+\infty} \frac{1}{\zeta^k} = +\infty \). Then \( \liminf_{k \to +\infty} \|g_k\| = 0 \).

**V. SIMULATION RESULTS**

In this section, in order to evaluate the performance of Algorithm 1, which includes VLRGSL_{1/2} and MSE+VLRGSL_{1/2}, it will be compared with the gradient-based algorithm with

---

**FIGURE 4.** The numerical performance of the algorithms for Sonar problem with \( \eta = 0.005 \) and \( \kappa = 0.09 \).
the smoothing \( L_{1/2} \) regularization [3], denoted as GSL\(_{1/2} \), and Algorithm A3 with the best numerical performance in [22] on the same examples: identification of the function \( y = \sin(\pi x) \), and a Sonar classification problem acquired from UCI repository [26]. All simulations are performed with Matlab 2018b on a computer (Intel (R) Core (TM) i5-8500 CPU @ 3.00 GHz 3.00 GHz 8.00 GB RAM). For algorithms, the initial fuzzy parameters are chosen stochastically in \([0, 1] \), the maximum epochs \( K_{\text{max}} \) is 10,000, multiple values of the learning rate \( \eta \) and the penalty parameter \( \lambda \) are tested, 10 trials are carried out, and 10 fuzzy rules are used. In order to estimate \( L_k \), we set \( L_0 = 1 \) and \( c = 0.1 \) in (14), and \( r = 50 \) and \( \mu = 0.5 \) in (17).

In the following tables, ACAN (defined in [6]) denotes the average number of identified zero coefficients or below a certain threshold and the threshold is set to 0.0001, Std means the standard deviation of errors over multiple runs, and the best performance of the algorithms are highlighted by boldface except for Average of time.

### A. IDENTIFICATION PROBLEM

Identification of the nonlinear function \( y = \sin(\pi x) \), \( x \in [-1, 1] \). In this example, training patterns are evenly selected from the interval \( x \in [-1, 1] \) with 0.02 as the discretization size. Test patterns are selected with \( \frac{1}{15} \) as the discretization size similarly.

The numerical results are presented in Table 1. It shows that the whole performance of VLRGSL\(_{1/2} \) and MSE+VLRGSL\(_{1/2} \) are superior to that of GSL\(_{1/2} \) and Algorithm A3 from the 4th-6th columns, and MSE+VLRGSL\(_{1/2} \) can usually obtain the lowest error and have the most robust performance. GSL\(_{1/2} \) usually costs the least time due to the constant learning rate. For Algorithm A3, it seems that the algorithm with the smaller constant (\( \eta = 0.001 \)) has the poor performance, and also is shown in Figure 3.

In Figure 2, when \( \lambda \) is a bit large, the curve of the norm of the gradient on GSL\(_{1/2} \) shows serious oscillation, and for VLRGSL\(_{1/2} \), the curve also occurs oscillation. However, by using the modified secant equation (15) in (17),...
MSE+VLRGSL$_{1/2}$ greatly reduces oscillation compared with VLRGSL$_{1/2}$.

**B. THE SONAR CLASSIFICATION PROBLEM**

The Sonar benchmark problem is a famous binary classification problem. It needs to classify reflected sonar signals in two categories (metal cylinders and rocks). The data set is composed by 208 input vectors, and each vector has 60 components. For input vectors, we stochastically select 128 input vectors from the data set as training patterns and the initial connecting weights in $[0, 1]$. The correct identification on vectors from the data set as training patterns and the initial connecting weights is proved. We will also investigate the corresponding stochastic algorithm for large scale fuzzy systems in the future research.

**VI. CONCLUSION**

A variant of the common batch gradient descent algorithm with the smoothing $L_{1/2}$ regularization is proposed to train the zero-order T-S inference network. Utilizing a new adjust scheme of the estimation of Lipschitz constant and the modified secant equation, the proposed algorithm makes better use of the second-order information to reduce the adverse effects of high non-linearity and ill-condition of the error function. Simulation results show that the proposed algorithm has better numerical performance, sparsity and generalization capacity. Moreover, under the assumptions, the convergent result is proved. We will also investigate the corresponding stochastic algorithm for large scale fuzzy systems in the future research.

**REFERENCES**

[1] W. Wu, Q. Fan, J. M. Zurada, J. Wang, D. Yang, and Y. Liu, “Batch gradient method with smoothing $L_{1/2}$ regularization for training of feedforward neural networks,” *Neural Netw.*, vol. 50, pp. 72–78, Feb. 2014.

[2] Q. Fan, J. M. Zurada, and W. Wu, “Convergence of online gradient method for feedforward neural networks with smoothing $L_{1/2}$ regularization penalty,” *Neurocomputing*, vol. 131, pp. 208–216, May 2014.

[3] Y. Liu, W. Wu, Q. Fan, D. Yang, and J. Wang, “A modified gradient learning algorithm with smoothing $L_{1/2}$ regularization for Takagi–Sugeno fuzzy models,” *Neurocomputing*, vol. 138, pp. 229–237, Aug. 2014.

[4] W. Wu and J. Yang, “$L_{1/2}$ regularization methods for weights sparsification of neural networks,” (in Chinese), *Sci. Sin. Math.*, vol. 45, pp. 1–18, Aug. 2015.

[5] W. Li, Y. Liu, J. Yang, and W. Wu, “A new conjugate gradient method with smoothing $L_{1/2}$ regularization based on a modified secant equation for training neural networks,” *Neural Process. Lett.*, vol. 48, no. 2, pp. 955–978, Oct. 2018.

[6] Z. B. Xu, H. Zhang, Y. Wang, and X. Y. Chang, “$L_{1/2}$ regularizer,” *Sci. China Inf. Sci.*, vol. 53, no. 6, pp. 1159–1169, May 2010.

[7] V. Boskovitz and H. Gutfman, “An adaptive neuro-fuzzy system for automatic image segmentation and edge detection,” *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 247–262, Apr. 2002.

[8] R. Babuška and H. Verbruggen, “Neuro-fuzzy methods for nonlinear system identification,” *Annu. Rev. Control*, vol. 27, no. 1, pp. 73–85, Jan. 2003.

[9] A. Subasi, “Application of adaptive neuro-fuzzy inference system for epileptic seizure detection using wavelet feature extraction,” *Comput. Biol. Med.*, vol. 37, no. 2, pp. 227–244, Feb. 2007.

[10] S. Kar, S. Das, and P. K. Ghosh, “Applications of neuro fuzzy systems: A brief review and future outline,” *Appl. Soft Comput.*, vol. 15, pp. 243–259, Feb. 2014.

[11] S. Stoeva and A. Nikov, “A fuzzy backpropagation algorithm,” *Fuzzy Sets Syst.*, vol. 112, no. 1, pp. 27–39, May 2000.

[12] A. Nikov and S. Stoeva, “Quick fuzzy backpropagation algorithm,” *Neural Netw.*, vol. 14, no. 2, pp. 231–244, Mar. 2001.

[13] A. E. Ruano, C. Cabrita, J. V. Oliveira, and L. T. Kóczy, “Supervised training algorithms for B-Spline neural networks and neuro-fuzzy systems,” *Int. J. Syst. Sci.*, vol. 33, no. 8, pp. 689–711, Jan. 2002.

[14] W. Yu and X. Li, “Fuzzy identification using fuzzy neural networks with stable learning algorithms,” *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 3, pp. 411–420, Jun. 2004.
[15] W. Wu, L. Li, J. Yang, and Y. Liu, “A modified gradient-based neuro-fuzzy learning algorithm and its convergence,” Inf. Sci., vol. 180, no. 9, pp. 1630–1642, May 2010.

[16] Y. Liu and D. Yang, “Convergence analysis of the batch gradient-based neuro-fuzzy learning algorithm with smoothing $L_{1/2}$ regularization for the first-order Takagi–Sugeno system,” Fuzzy Sets Syst., vol. 319, pp. 28–49, Jul. 2017.

[17] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, “Accelerating the convergence of the back-propagation method,” Biol. Cybern., vol. 59, nos. 4–5, pp. 257–263, Sep. 1988.

[18] C. Igel and M. Hüsken, “Empirical evaluation of the improved Rprop learning algorithms,” Neurocomputing, vol. 50, pp. 105–123, Jan. 2003.

[19] H. Shao and G. Zheng, “Convergence analysis of a back-propagation algorithm with adaptive momentum,” Neurocomputing, vol. 74, no. 5, pp. 749–752, Feb. 2011.

[20] I. E. Livieris and P. Pintelas, “A new conjugate gradient algorithm for training neural networks based on a modified secant equation,” Appl. Math. Comput., vol. 221, pp. 491–502, Sep. 2013.

[21] J. Sun and J. Zhang, “Global convergence of conjugate gradient methods without line search,” Ann. Oper. Res., vol. 103, nos. 1–4, pp. 161–173, 2001.

[22] Z.-J. Shi and J. Shen, “Step-size estimation for unconstrained optimization methods,” Comput. Appl. Math., vol. 24, no. 3, pp. 399–416, Dec. 2005.

[23] I. del Campo, J. Echanobe, G. Bosque, and J. M. Tarela, “Efficient hardware/software implementation of an adaptive neuro-fuzzy system,” IEEE Trans. Fuzzy Syst., vol. 16, no. 3, pp. 761–778, Jun. 2008.

[24] H. Ichihashi and I. B. Türksen, “A neuro-fuzzy approach to data analysis of pairwise comparisons,” Int. J. Approx. Reasoning, vol. 9, no. 3, pp. 227–248, Oct. 1993.

[25] D.-H. Li and M. Fukushima, “A modified BFGS method and its global convergence in nonconvex minimization,” J. Comput. Appl. Math., vol. 129, nos. 1–2, pp. 15–35, Apr. 2001.

[26] C. Blake and C. Merz. (1998). UCI Repository of Machine Learning Database. [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository

[27] N. Ampazis and S. J. Perantonis, “Two highly efficient second-order algorithms for training feedforward networks,” IEEE Trans. Neural Netw., vol. 13, no. 5, pp. 1064–1074, Sep. 2002.

**YUNLONG LU** received the B.S. degree in applied mathematics from Beihua University, Jilin, China, in 2002, the M.S. degree in operational research and cybernetics from the Harbin Institute of Technology, Harbin, China, in 2006, and the Ph.D. degree in computational mathematics from the Dalian University of Technology, China, in 2019. He is currently an Associate Professor with the School of Mathematics and Statistics, Beihua University. His research interests include numerical optimization and machine learning.

**WENYU LI** received the B.S. and M.S. degrees in applied mathematics from Beihua University, Jilin, China, in 2005 and 2008, respectively, and the Ph.D. degree in computational mathematics from the Dalian University of Technology, China, in 2018. She is currently an Associate Professor with the School of Mathematics and Statistics, Beihua University. Her research interests include numerical optimization and the learning methods of neural networks.

**HONGWEI WANG** received the B.S. degree in mathematics from the Chongqing Technology and Business University, Chongqing, China, in June 2012, and the M.S. and Ph.D. degrees in mathematics from Texas Tech University, Lubbock, Texas, in May 2015 and December 2017, respectively. She is currently an Assistant Professor with the Department of Mathematics and Physics, Texas A&M International University, Laredo, Texas. Her research interests include knot theory in topology, analysis of college students’ health, and children overweight and obesity in South Texas.