Neural Networks 89 (2017) 19–30

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Fractional-order gradient descent learning of BP neural networks with Caputo derivative☆

Jian Wang [a,*], Yanqing Wen [a], Yida Gou [a], Zhenyun Ye [b], Hua Chen [a,*]

[a] College of Science, China University of Petroleum, Qingdao, 266580, China

[b] College of Computer & Communication Engineering, China University of Petroleum, Qingdao, 266580, China

ARTICLE INFO

Article history:
Received 22 June 2016
Received in revised form 4 February 2017
Accepted 14 February 2017
Available online 22 February 2017

Keywords:
Fractional calculus
Backpropagation
Caputo derivative
Monotonicity
Convergence

ABSTRACT

Fractional calculus has been found to be a promising area of research for information processing and modeling of some physical systems. In this paper, we propose a fractional gradient descent method for the backpropagation (BP) training of neural networks. In particular, the Caputo derivative is employed to evaluate the fractional-order gradient of the error defined as the traditional quadratic energy function. The monotonicity and weak (strong) convergence of the proposed approach are proved in detail. Two simulations have been implemented to illustrate the performance of presented fractional-order BP algorithm on three small datasets and one large dataset. The numerical simulations effectively verify the theoretical observations of this paper as well.

1. Introduction

Fractional differential calculus has been a classical notion in mathematics for several hundreds of years. It is based on differentiation and integration of arbitrary fractional order, and as such it is a generalization of the popular integer calculus. Yet only recently it has been applied to the successful modeling of certain physical phenomena. A number of papers in the literature (Kvitsinskii, 1993; Love, 1971; McBride, 1986; Miller, 1995; Nishimoto, 1989; Oldham & Spanier, 1974) have recently reported fractional differential calculus as a model which describes much better than the conventional integer calculus on some certain selected natural phenomena. Descriptions of viscosity of liquids, diffusion of EM waves and fractional kinetics are such a few examples of dynamics of certain systems that can be successfully expressed in fractional formats. A good literature review in support of this statement is in Wang, Yu, Wen, Zhang, and Yu (2015).

As a consequence, the study of dynamics based on fractional-order differential systems has attracted considerable research interest. Novel results include solutions for chaos and stability analysis in fractional-order systems (Delavari, Baleanu, & Sadati, 2012; Deng & Li, 2005; Wu, Li, & Chen, 2008). In Wu et al. (2008), fractional multipoles, fractional solutions for Helmholtz, and fractional image processing methods were effectively studied and promising results have been produced in electromagnetics. By using the Laplace transform theory, chaos synchronization of the fractional Lü system with suitable conditions has been meticulously proved (Deng & Li, 2005). In Delavari et al. (2012), the Lyapunov direct method was extended to describe the Caputo type fractional-order nonlinear systems, and the comparison theorem of these systems was proposed by using Bihari's and Bellman–Gronwall's inequalities.

The fractional differential calculus has also been successfully adopted by the field of neural networks. Some remarkable research of fractional-order neural networks has been presented in Chen (2013), Chen and Chen (2016), Rakkiyappan, Cao, and Velmurugan (2015); Rakkiyappan, Sivaranjani, Velmurugan, and Cao (2016) and Zhang, Yu, and Wang (2015). In Chen (2013), fractional calculus was used for the Backpropagation (BP) (Rumelhart, Hinton, & Williams, 1986) algorithm for feedforward neural networks (FNNs). The simulation results demonstrated that the convergence speed based on fractional-order FNNs was much faster than integer-order FNNs. By extending the second method of Lyapunov,

☆ This work was 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 (Nos. ZR2013FQ004, ZR2013DM015), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20130133120014) and the Fundamental Research Funds for the Central Universities (Nos. 13CX05016A, 14CX05042A, 15CX05053A, 15CX08011A).

* Corresponding authors.
E-mail addresses: wangjiannl@upc.edu.cn (J. Wang), chenhua@upc.edu.cn (H. Chen).

http://dx.doi.org/10.1016/j.neunet.2017.02.007
0893-6080/

the Mittag-Leffler stability analysis was performed for fractional-order Hopfield neural networks (Zhang et al., 2015). In Zhang et al. (2015), the stability conditions have been used to achieve complete and quasi synchronization in the coupling case of these networks with constant or time-dependent external inputs. The global Mittag-Leffler stability and global asymptotical periodicity of the fractional-order non-autonomous neural networks were successfully investigated in Chen and Chen (2016) by using a fractional-order differential and integral inequality technique. For fractional-order complex-valued neural networks, the existence and stability analyses have been studied in detail in Rakkiyappan et al. (2015, 2016). In Xiao, Zheng, Jiang, and Cao (2015), a fractional-order recurrent neural network model was studied with commensurate or incommensurate orders to exhibit the dynamic behaviors. The simulation results demonstrate that the dynamics of fractional-order systems are not invariant, in contrast to integer-order systems.

However, most research findings for fractional-order systems have been limited to studies of fully coupled recurrent networks of Hopfield type (Chen & Chen, 2016; Rakkiyappan et al., 2015, 2016; Wang, Yu, & Wen, 2014; Wu & Zen, 2013; Wu, Zhang, & Zen, 2011a; Xiao et al., 2015; Zhang et al., 2015). The vast majority of papers have been focused on studying properties of fixed points for non-integer order differential equations that describe such networks. The researched networks vary in their properties: they are with or without delay in the feedback loop, while other extensions have provided generalizations to complex-valued neurons. In contrast, this work concerns fractional-order error BP in FNNs.

Gradient descent method is commonly used to train FNNs by minimizing the error function, being the norm of a distance between the actual network output and the desired output. There exist other optional methods to implement the BP algorithm for FNNs, such as conjugate gradient, Gauss–Newton and Levenberg–Marquardt. We note that all of the above optimal methods are typically employed to train integer-order FNNs.

To our best knowledge, there is very limited literature focused on the convergence analysis of fractional-order FNNs. The existing convergence results are mainly concentrated on integer-order gradient-based FNNs. For batch mode training, the BP algorithm of FNNs with penalty was proven to be deterministic convergent (Wu, Shao, & Li, 2006). In addition, the weight sequence is uniformly bounded due to the influence of the penalty term. For incremental training, weak and strong convergence results are obtained with or without penalty terms based on different assumptions for the activation function, learning rates and the stationary points of the objective function (Shao, Wu, & Liu, 2010; Wu, Wang, Cheng, & Li, 2011b; Xu, Zhang, & Jin, 2009).

Only very recently, fractional neural networks have been evaluated in a broader context of training and the minimization of an objective function (Pu et al., 2015). This paper has offered a detailed analysis of fractional gradient descent learning conditions, and has supported it with initial convergence studies of minimization of quadratic energy norm and also with numerical illustration of searching for extreme points during the Fractional Adaptive Learning (FAL).

Inspired by Chen (2013) and Pu et al. (2015), we apply the fractional steepest descent algorithm to train FNNs. In particular, we employ the Caputo derivative formula to compute the fractional-order gradient of the error function with respect to the weights and obtain the deterministic convergence. The main contributions of this paper are as follows:

(1) Under suitable conditions for activation functions and the learning rate, the monotonic decreasing property of the error function has been guaranteed.
For the activation functions of hidden and output layers, we assume that the first and second derivatives of the activation functions are uniformly bounded. This condition can be easily satisfied since the most common sigmoid activation functions are uniformly bounded on $\mathbb{R}$ and infinitely differentiable.
(2) The deterministic convergence of the proposed fractional-order training algorithm has been rigorously proved, which prevents the divergence behavior from a theoretical point of view. The weak convergence means that the fractional-order gradient of the error function approaches zero when the iteration number tends to infinity, while the strong convergence means the weight sequence goes to a fixed point.
(3) Numerical simulations are reported to illustrate the effectiveness of the proposed factional-order neural networks and support the theoretical results in this paper.

Selected benchmark UCI datasets have been used to compare the performances of fractional-order vs. integer-order based FNNs. The simulations demonstrate that the training and testing accuracies for fractional-order FNNs are better than those for first-order gradient based FNNs. In addition, the monotonicity of error function and weak convergence have been figured out through the MNIST dataset.

We note that the error function in Pu et al. (2015) is quite different from that in this paper. In Pu et al. (2015), the error function is defined as the sum of the first-order extreme values of the error function and the quadratic norm of the parameters and its first-order extreme value. In this work, we consider the error function to be the squared error being the difference between the actual output and the desired output. Another difference between (Pu et al., 2015) and this paper is in the definition of the convergence itself. In Pu et al. (2015), it focuses on the regular convergence rate of the proposed algorithm, that is, on the exponential convergence. In this paper the convergence behavior of fractional-order BP algorithm is evaluated. That is, the norm of the gradient of the error function approaches zero (weak convergence). Authors also provide strong convergence proof.

The structure of the paper is as follows: in Section 2, the definitions of three commonly used fractional-order derivative are introduced. The traditional BP algorithm and our novel algorithm of fractional-order BP neural networks training based on Caputo derivative are presented in Section 3. In Section 4, the convergence results are presented, and the detailed proofs of the main convergence results are stated as Appendix. Numerical simulations are presented to illustrate the effectiveness of our results in Section 5. Finally, the paper is concluded in the last section.

2. Fractional-order derivative

Unlike the situation with integer-order derivatives, several definitions are used for fractional-order derivatives. The three most common definitions are by Grunwald–Letnikov (GL), Riemann–Liouville (RL), and Caputo (Love, 1971; McBride, 1986; Nishimoto, 1989; Oldham & Spanier, 1974).

Definition 2.1 (*GL Fractional-Order Derivative*)**.** The GL derivative with order α of function $f(t)$ is defined as

$$ {}^{GL}_{a}D^{\alpha}_{t}f(t) = \sum_{k=0}^{n} \frac{f^{(k)}(a)(t-a)^{-\alpha+k}}{\Gamma(-\alpha+k+1)} + \frac{1}{\Gamma(n-\alpha+1)} \int_{a}^{t} (t-\tau)^{n-\alpha} f^{(n+1)}(\tau) d\tau, \quad (1) $$

where ${}^{GL}_{a}D^{\alpha}_{t}$ is the GL fractional derivative operator, $\alpha > 0$, $n-1 < \alpha < n$, $n \in N$, $f(t)$ is the objective function under consideration and $[a, t]$ is the interval of $f(t)$. $\Gamma(\cdot)$ is the Gamma function, that is, $\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t} dt$.

Definition 2.2 (*RL Fractional-Order Derivative*)**.** The Riemann–Liouville derivative with order α of function $f(t)$ is defined as

$$
\begin{aligned}
{}^{RL}{}_aD_t^{\alpha} f(t) &= \frac{d^n}{dt^n}\, {}_aD_t^{-(n-\alpha)} f(t) \\
&= \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t (t-\tau)^{n-\alpha-1} f(\tau) d\tau, \qquad (2)
\end{aligned}
$$

where ${}^{RL}{}_aD_t^{\alpha}$ is the RL fractional derivative operator. Moreover, the GL fractional derivative can be deduced from the definition of the RL fractional derivative.

Definition 2.3 (*Caputo Fractional-Order Derivative*)**.** The definition of the Caputo fractional-order derivative of order α is defined as follows

$$
{}^{Caputo}{}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t (t-\tau)^{n-\alpha-1} f^{(n)}(\tau) d\tau, \qquad (3)
$$

where ${}^{Caputo}{}_aD_t^{\alpha}$ is the Caputo derivative operator, α is the fractional order.

Particularly, when $\alpha \in (0, 1)$, the expression for Caputo derivative is as follows

$$
{}^{Caputo}{}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(1-\alpha)} \int_a^t (t-\tau)^{-\alpha} f'(\tau) d\tau. \qquad (4)
$$

When $\alpha \in (0, 1)$ and $a = 0$, there is a visible difference in the derivation with respect to a constant between RL derivative and Caputo derivative. Suppose that K is a constant, it is easy to conclude that ${}^{RL}{}_aD_t^{\alpha} K = \frac{K}{\Gamma(1-\alpha)} x^{-\alpha} \neq 0$ and ${}^{Caputo}{}_aD_t^{\alpha} K = 0$.

Since the initial value of fractional differential equations with the Caputo derivative is the same as that of the integer differential equations: this derivative has more applications in physical processes and engineering problems (Gorenflo & Mainardi, 2008). In this paper, we just employ the Caputo fractional-order derivative to evaluate the BP training algorithm for fractional FNNs. For convenience, we use the notion ${}_aD_t^{\alpha}$ to denote Caputo fractional derivative operator instead of ${}^{Caputo}{}_aD_t^{\alpha}$.

3. Algorithm description

Let us begin with an introduction of a network with three layers. The numbers of neurons for the input, hidden and output layers are p, n and 1, respectively. Suppose that the training sample set is $\{\mathbf{x}^j, O^j\}_{j=1}^J \subset \mathbb{R}^p \times \mathbb{R}$, where $\mathbf{x}^j$ and O^j are the input and the corresponding ideal output of the jth sample. Let $g, f : \mathbb{R} \to \mathbb{R}$ be given activation functions for the hidden and the output layers, separately. Let $\mathbf{V} = (v_{ij})_{n\times p}$ be the weight matrix connecting the input and the hidden layers, and write $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{ip})^T \in \mathbb{R}^p$ for $i = 1, 2, \ldots, n$. The weight vector connecting the hidden and output layers is denoted by $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T \in \mathbb{R}^n$. To simplify the presentation, we combine the weight matrix $\mathbf{V}$ with the weight vector $\mathbf{u}$, and write the total weight vector as follows

$$
\mathbf{w} = (\mathbf{u}, \mathbf{V}) = (\mathbf{u}^T, \mathbf{v}_1^T, \ldots, \mathbf{v}_n^T)^T \in \mathbb{R}^{n(p+1)}. \qquad (5)
$$

For convenience, we introduce the following vector valued function

$$
G(\mathbf{z}) = (g(z_1), g(z_2), \ldots, g(z_n))^T, \quad \forall \mathbf{z} \in \mathbb{R}^n. \qquad (6)
$$

3.1. BP algorithm based on gradient descent method

For any given jth input $\mathbf{x}^j \in \mathbb{R}^p$, the final actual output is

$$
y = f\left(\mathbf{u} \cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right). \qquad (7)
$$

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

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2} \sum_{j=1}^{J} \left(O^j - f\left(\mathbf{u} \cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right)\right)^2 \\
&= \sum_{j=1}^{J} f_j(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)), \qquad (8)
\end{aligned}
$$

where $f_j(t) = \frac{1}{2}(O^j - f(t))^2, j = 1, 2, \ldots, J, t \in \mathbb{R}$. We note that the constructed function $f_j(\cdot)$ is a composite function of f and G in terms of the jth sample.

Given an initial weight $\mathbf{w}^0 = \left(\mathbf{u}^0, \mathbf{V}^0\right)$, the kth weight vector, $\mathbf{w}^k = \left(\mathbf{u}^k, \mathbf{V}^k\right)$, is iteratively updated according to the classical BP training algorithm (Rumelhart et al., 1986). To be specific, we show the updating formula with the following forms,

$$
u_i^{k+1} = u_i^k - \eta E_{u_i^k}(\mathbf{w}), \qquad (9)
$$

$$
v_{im}^{k+1} = v_{im}^k - \eta E_{v_{im}^k}(\mathbf{w}), \qquad (10)
$$

where $k \in \mathbb{N}, i = 1, 2, \ldots, n, m = 1, 2, \ldots, p$, u_i^k and v_{im}^k are the corresponding component weights of the weight vector $\mathbf{u}^k$ and $\mathbf{v}_i^k$, respectively; $\eta > 0$ is the learning rate; and

$$
E_{u_i^k}(\mathbf{w}) = \sum_{j=1}^{J} f_j'(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)) g(\mathbf{v}_i^k \cdot \mathbf{x}^j), \qquad (11)
$$

$$
E_{v_{im}^k}(\mathbf{w}) = \sum_{j=1}^{J} f_j'(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)) u_i g'(\mathbf{v}_i^k \cdot \mathbf{x}^j) x_m^j, \qquad (12)
$$

where $k \in \mathbb{N}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, J$.

3.2. Fractional-order BP algorithm based on Caputo fractional-order derivative

For the given jth input sample $\mathbf{x}^j \in \mathbb{R}^p$, $\theta^{i,j} = v_{i1}x_1{}^j + v_{i2}x_2{}^j + \cdots + v_{ip}x_p{}^j = \mathbf{v}_i \cdot \mathbf{x}^j$ $(i = 1, 2, \ldots, n)$ is the jth input value of ith hidden neuron. The input of output layer is represented by $\zeta^j = u_1 g(\mathbf{v}_1 \cdot \mathbf{x}^j) + u_2 g(\mathbf{v}_2 \cdot \mathbf{x}^j) + \cdots + u_n g(\mathbf{v}_n \cdot \mathbf{x}^j) = \mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)$ and the actual output is given by $y^j = f(\mathbf{u} \cdot G\left(\mathbf{V}\mathbf{x}^j\right))$ after activation.

For any fixed weights $\mathbf{w}$, a conventional square error function is given as

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2} \sum_{j=1}^{J} \left(O^j - f\left(\mathbf{u} \cdot G\left(\mathbf{V}\mathbf{x}^j\right)\right)\right)^2 \\
&= \sum_{j=1}^{J} f_j(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)), \qquad (13)
\end{aligned}
$$

where $f_j(t) = \frac{1}{2}(O^j - f(t))^2, j = 1, 2, \ldots, J, t \in \mathbb{R}$.

For any initial weight $\mathbf{w}^0 = (\mathbf{u}^0, \mathbf{V}^0)$, the general kth weight vector $\mathbf{w}^k = \left(\mathbf{u}^k, \mathbf{V}^k\right)$ of weight sequence is adjusted in terms of Caputo α-order steepest descent method. To be specific, the updating formulas of the components of $\mathbf{w}^k$ are given as follows

$$
u_i^{k+1} = u_i^k - \eta_c D_{u_i^k}^{\alpha} E(\mathbf{w}), \qquad (14)
$$

$$
v_{im}^{k+1} = v_{im}^k - \eta_c D_{v_{im}^k}^{\alpha} E(\mathbf{w}), \qquad (15)
$$

where $\eta > 0$ is the learning rate, $0 < \alpha < 1$ is the fractional order and $c = \min\{u_i^k, v_{im}^k\} (k \in \mathbb{N}, i = 1, \ldots, n, m = 1, \ldots, p)$.

According to the definition of Caputo fractional derivative, the fractional-order differential of a composite function may be represented as the product of an integer-order differential and a fractional order differential. Suppose $h(s(t))$ to be a composite function, its α-order derivative with respect to t is

$$ {}_aD_t^\alpha h(s) = \frac{\partial}{\partial s}(h(s)) \cdot {}_aD_t^\alpha s(t). \tag{16} $$

Next, we will divide it into two parts:

Part 1. ${}_cD_{u_i^k}^\alpha E(\mathbf{w})$.

According to (16), we have

$$ {}_cD_{u_i^k}^\alpha E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \zeta^j} \cdot {}_cD_{u_i^k}^\alpha(\zeta^j), \tag{17} $$

where

$$ \frac{\partial E(\mathbf{w})}{\partial \zeta^j} = \sum_{j=1}^{J} f_j'(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)). \tag{18} $$

By the definition of Caputo α-order derivative, $0 < \alpha < 1$, we see that

$$ \begin{aligned} {}_cD_{u_i^k}^\alpha(\zeta^j) &= \frac{1}{\Gamma(1-\alpha)} \int_c^{u_i^k} (u_i^k - \tau)^{-\alpha} g(\mathbf{v}_i^k \cdot \mathbf{x}^j) d\tau \\ &= \frac{1}{\Gamma(1-\alpha)} g(\mathbf{v}_i^k \cdot \mathbf{x}^j) \int_c^{u_i^k} (u_i^k - \tau)^{-\alpha} d\tau \\ &= \frac{1}{\Gamma(1-\alpha)} \frac{1}{1-\alpha} g(\mathbf{v}_i^k \cdot \mathbf{x}^j)(u_i^k - c)^{1-\alpha}, \end{aligned} \tag{19} $$

where $k \in \mathbb{N}, i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, J$.

Therefore, it follows from (17)–(19) that

$$ \begin{aligned} {}_cD_{u_i^k}^\alpha E(\mathbf{w}) &= \frac{1}{(1-\alpha)\Gamma(1-\alpha)} \\ &\quad \times \sum_{j=1}^{J} f_j'(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)) g(\mathbf{v}_i^k \cdot \mathbf{x}^j)(u_i^k - c)^{1-\alpha}. \end{aligned} \tag{20} $$

Part 2. ${}_cD_{v_{im}^k}^\alpha E(\mathbf{w})$.

Similar to (17), we deduce that

$$ {}_cD_{v_{im}^k}^\alpha E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \theta^{i,j}} \cdot {}_cD_{v_{im}^k}^\alpha(\theta^{i,j}), \tag{21} $$

where

$$ \frac{\partial E(\mathbf{w})}{\partial \theta^{i,j}} = \sum_{j=1}^{J} f_j'(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)) u_i g'(\mathbf{v}_i \cdot \mathbf{x}^j). \tag{22} $$

Employing (4), we find that

$$ \begin{aligned} {}_cD_{v_{im}^k}^\alpha(\theta^{i,j}) &= \frac{1}{\Gamma(1-\alpha)} \int_c^{v_{im}^k} (v_{im}^k - \tau)^{-\alpha} x_m^j d\tau \\ &= \frac{1}{\Gamma(1-\alpha)} x_m^j \left(\frac{1}{1-\alpha} (v_{im}^k - \tau)^{1-\alpha} \Big|_c^{v_{im}^k} \right) \\ &= \frac{1}{(1-\alpha)\Gamma(1-\alpha)} x_m^j (v_{im}^k - c)^{1-\alpha} \end{aligned} \tag{23} $$

where $k \in \mathbb{N}, i = 1, 2, \ldots, n, m = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, J$. Combining (21)–(23), we have that

$$ \begin{aligned} {}_cD_{v_{im}^k}^\alpha E(\mathbf{w}) &= \frac{1}{(1-\alpha)\Gamma(1-\alpha)} \\ &\quad \times \sum_{j=1}^{J} f_j'(\mathbf{u} \cdot G(\mathbf{V}\mathbf{x}^j)) u_i^k g'(\mathbf{v}_i \cdot \mathbf{x}^j) x_m^j (v_{im}^k - c)^{1-\alpha}, \end{aligned} \tag{24} $$

where $k \in \mathbb{N}, i = 1, 2, \ldots, n, m = 1, 2, \ldots, p, j$ represents the order of the sample and the parameter α is the fractional order of Caputo derivative.

To be simple, we write the vector form of the above fractional-order Caputo derivative

$$ \begin{aligned} {}_cD_{\mathbf{w}}^\alpha E(\mathbf{w}) &= ({}_cD_{u_1}^\alpha E(\mathbf{w}), \ldots, {}_cD_{u_n}^\alpha E(\mathbf{w}), {}_cD_{v_{11}}^\alpha E(\mathbf{w}), \\ &\quad \ldots, {}_cD_{v_{np}}^\alpha E(\mathbf{w})). \end{aligned} \tag{25} $$

4. Main results

In this section, the convergent behavior of the proposed Caputo fractional-order BP training algorithm is described. For any $\mathbf{x} = (x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$, we write $\|\mathbf{x}\| = \sqrt{\sum_{m=1}^p x_m^2}$, where $\|\cdot\|$ stands for the Euclidean norm in $\mathbb{R}^p$. Let $\widehat{\Phi} = \{\mathbf{w} \in \Phi : {}_cD_{\mathbf{w}}^\alpha E(\mathbf{w}) = 0\}$ be the α-order stationary point set of the error function $E(\mathbf{w})$, where $\Phi \subset \mathbb{R}^{n(p+1)}$ is a bounded open set and $0 < \alpha < 1$.

Actually, we assume that the activation functions g and f are bounded and infinitely differentiable on $\mathbb{R}$, furthermore, all of its corresponding derivatives are also continuous and bounded on $\mathbb{R}$. For the convenience of theoretical analysis, we assume that the uniform upper bound of $|g(t)|, |g'(t)|, |g''(t)|, |f_j'(t)|$ and $|f_j''(t)|$ $(j = 1, \ldots, J)$ is C_1. In terms of the assumption (A2) and the finite samples, we denote that

$$ C_2 \triangleq \max_{1 \le j \le J} \|\mathbf{x}^j\|, \tag{26} $$

$$ C_3 \triangleq \sup_{k \in \mathbb{N}} \|\mathbf{w}^k\|, \tag{27} $$

where $j = 1, \ldots, J, k \in \mathbb{N}$. An upper bounded assumption of learning rate is an essential part to guarantee the deterministic convergence of the proposed algorithm. It is shown as below

$$ \gamma = \frac{(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}}{\frac{1}{2}nJC_1^2C_3C_2^2 + \frac{C_1J}{2}(1 + C_1^2C_2^2) + nJC_1^3 + JC_1^3C_2^2C_3^2}, \tag{28} $$

where the parameters $\alpha, c, n, J, C_1, C_2, C_3$ are already listed above.

To be clear, we list all of the following assumptions which are employed to analyze the theoretical results of the proposed algorithm.

(A1) The activations g and f are the common Sigmoid functions;
(A2) The boundedness of the weight sequence $\{\mathbf{w}^k\}$ $(k \in \mathbb{N})$ is valid during training procedure;
(A3) The learning rate η is a positive number with an upper bound γ;
(A4) The set $\widehat{\Phi}$ contains a finite number of points.

Theorem 4.1. *Suppose that the error function $E(\mathbf{w})$ is defined by (13), $\mathbf{w}^0 \in \mathbb{R}^{n(p+1)}$ be an arbitrary initial value, the learning sequence $\{\mathbf{w}^k\}$ be generated by (14) and (15). Assume that conditions (A1)–(A3) are valid, then we have*

(i) $E(\mathbf{w}^{k+1}) \le E(\mathbf{w}^k)$, $k \in \mathbb{N}$;
(ii) *There exists $E^* \ge 0$ such that* $\lim_{k\to\infty} E(\mathbf{w}^k) = E^*$;
(iii) $\lim_{k\to\infty} \|{}_cD_{\mathbf{w}}^\alpha E(\mathbf{w}^k)\| = 0$;
In addition, if assumption (A4) is also valid, then we have following strong convergence:
(iv) *There exists $\mathbf{w}^* \in \widehat{\Phi}$ such that* $\lim_{k\to\infty} \mathbf{w}^k = \mathbf{w}^*$.

Table 1
Datasets summary for classification.

Data set	Data size	Input features	Classes
1. Iris	150	4	3
2. Liver	345	6	2
3. Sonar	208	60	2

Table 2
Network structures and learning parameters for different datasets.

Data sets	Architecture	Initial interval	Max iteration	Learning rate
1. Iris	4-20-3	[−0.5,0.5]	600	0.5
2. Liver	6-22-2	[−0.5,0.5]	200	0.1
3. Sonar	60-20-2	[−0.8,0.8]	500	0.05

5. Experiments

To demonstrate the performance and the theoretical results of the presented algorithm, we carried out the following two simulations. The first simulation focuses on the comparison of the three different fractional-order BP algorithms: Caputo, RL and GL in Section 2. Specifically, the second one displays the observations of the proposed algorithm which based on the Caputo derivatives. In addition, the two curves of error function and the norm of gradient illustrate the theoretical conclusions of this paper. These two simulations have been carried out in MATLAB 2015 environment running an Intel i5, 2.67G CPU.

5.1. Three benchmark datasets

Due to the three classical definitions of fractional-order derivatives, it is necessary to compare their performance on benchmark datasets. The classification datasets listed in Table 1 for this simulation are three simple problems. Each dataset is randomly split into two subsets with a fixed proportion, training and testing samples are separately set to be 60% and 40%. To avoid the large inputs dominating the calculation, all of the datasets have been transformed into the interval $[-1, 1]$ in terms of preprocessing procedure.

For the above benchmark datasets, we construct the corresponding networks for the different learning algorithms (cf. Table 2). They differ by network structures, initial intervals of weights, maximum epochs and learning rates. For the sake of comparing the performance sufficiently, the simulations were repeated 5 times for each dataset. The 60% training samples of each dataset are first randomly chosen and then fixed during training for the different algorithms.

To analyze the numerical performance based on different fractional-order derivatives, three performance metrics have been conducted: training accuracy, testing accuracy and training time. Fig. 1 demonstrates the classification ability on training samples and the generalization on testing samples, separately. According to the three factional-order derivatives, the training and testing accuracies on Iris are respectively listed in Fig. 1 (a1) and (a2). It shows that the BP algorithm based on Caputo derivative performs much better than the other two algorithms. For Liver dataset, however, Fig. 1 (b1) and (b2) observe that the RL and GL based BP algorithms perform similar (except for the case of $\frac{1}{9}$-order) and better than Caputo based BP algorithm. While, it displays that Caputo based BP algorithm performs slightly better than GL and RL based BP algorithms on Sonar training samples in Fig. 1 (c1). Fig. 1 (c2) demonstrates that Caputo and RL based BP algorithms beat the GL based BP algorithm for the smaller fractional-order derivatives ($\frac{1}{9}$ and $\frac{2}{9}$ orders), and perform the similar generalization ability for the other fraction-orders. We note that all of these three algorithms perform very similar with each other when the fraction-order value approaches $\frac{8}{9}$ on both training and testing samples.

The training times listed in Table 3 represent the average running time for the given fractional-order and fractional derivatives. It clearly shows that the training procedure consumes pretty similar time for the different fractional orders in the same fractional-order derivative. Interestingly, for given fractional orders, the Caputo and GL based BP algorithms are more time consuming (approximately twice) than RL based BP algorithm. This would be an important topic to study the computation burden and reveal the inherent mechanism of these three different algorithms in the future.

5.2. MNIST

To verify the convergence of the proposed FAL-BP, simulations have been done on the MNIST handwritten digital dataset. The results have been compared with the common BP algorithm.

The MNIST handwritten digital dataset is collected from the NIST database of the National Institute of Standards and Technology. Digital images are normalized to an image 28×28, and represented as 784×1 vectors. Each element in the vector is a number between 0 and 255, representing the gray levels of each pixel. The MNIST database has a total number of 60,000 training samples and 10,000 testing samples. In this simulation, we employ different fractional α-order derivatives to compute the gradient of error function, where $\alpha = \frac{1}{9}, \frac{2}{9}, \frac{3}{9}, \frac{4}{9}, \frac{1}{2}, \frac{5}{9}, \frac{6}{9}, \frac{7}{9}, \frac{8}{9}$ and $\frac{9}{9} = 1$, separately ($\alpha = 1$ corresponds to standard integer-order derivative for the common BP). The networks have one hidden layer, and the architecture is set to be $784 \times \{100, 200, 300, 500\} \times 10$. For comparison, each network was trained 5 times to calculate the following average metrics: training accuracy, testing accuracy, training errors and norms of gradient of error function. For each training, the initial weights, stop criteria, learning rate and hidden nodes are chosen to be identical for the 10 different values of α, and the maximum training iteration is set to be 100 epochs.

The detailed codes are from (Nielsen, 2016). To speed up the training process, the codes were translated from Python to Matlab program language. Our programming algorithms are identical to those in Nielsen (2016).

To display the performances of differences of different fractional-order BP neural networks, we employed different learning parameters: learning rates and hidden nodes are set to be $\{0.5, 1, 2, 3\}$ and $\{100, 200, 300, 500\}$. For convenience, we graphed the training and testing accuracies with various learning rates for fixed number of hidden nodes in Fig. 2. Each row of the figure shows the training and testing accuracies based on various learning rates and fixed hidden nodes, that is, one can observe the tendencies of accuracy for each learning parameter. Generally speaking, the accuracies with larger learning rates are higher than those with smaller learning rates in each sub-figure. In addition, we observe that training and testing accuracies are gradually increasing with increasing fractional-orders and then reach the peak around $\frac{7}{9}$-order. One exception is in the sub-figure $b(2)$, the testing accuracy is slightly higher than that in $\frac{8}{9}$-order case (learning rate is 3). Another one is that the training accuracies of $\frac{6}{9}$-order network with 500 hidden nodes are equal to or slightly higher than those with $\frac{7}{9}$-order network.

Fig. 3 shows the performances of different fractional-order BP algorithms with different hidden nodes for each fixed learning rate. In this figure, each row displays the training and testing accuracies based on various hidden nodes and fixed learning rate. It helps to find variations of accuracy under different fractional-order derivatives. It is clear to see that the networks with 100 hidden nodes show lowest performance among various network architectures. In addition, we note that the other three networks with 200, 300 and 500 hidden nodes perform similarly. This

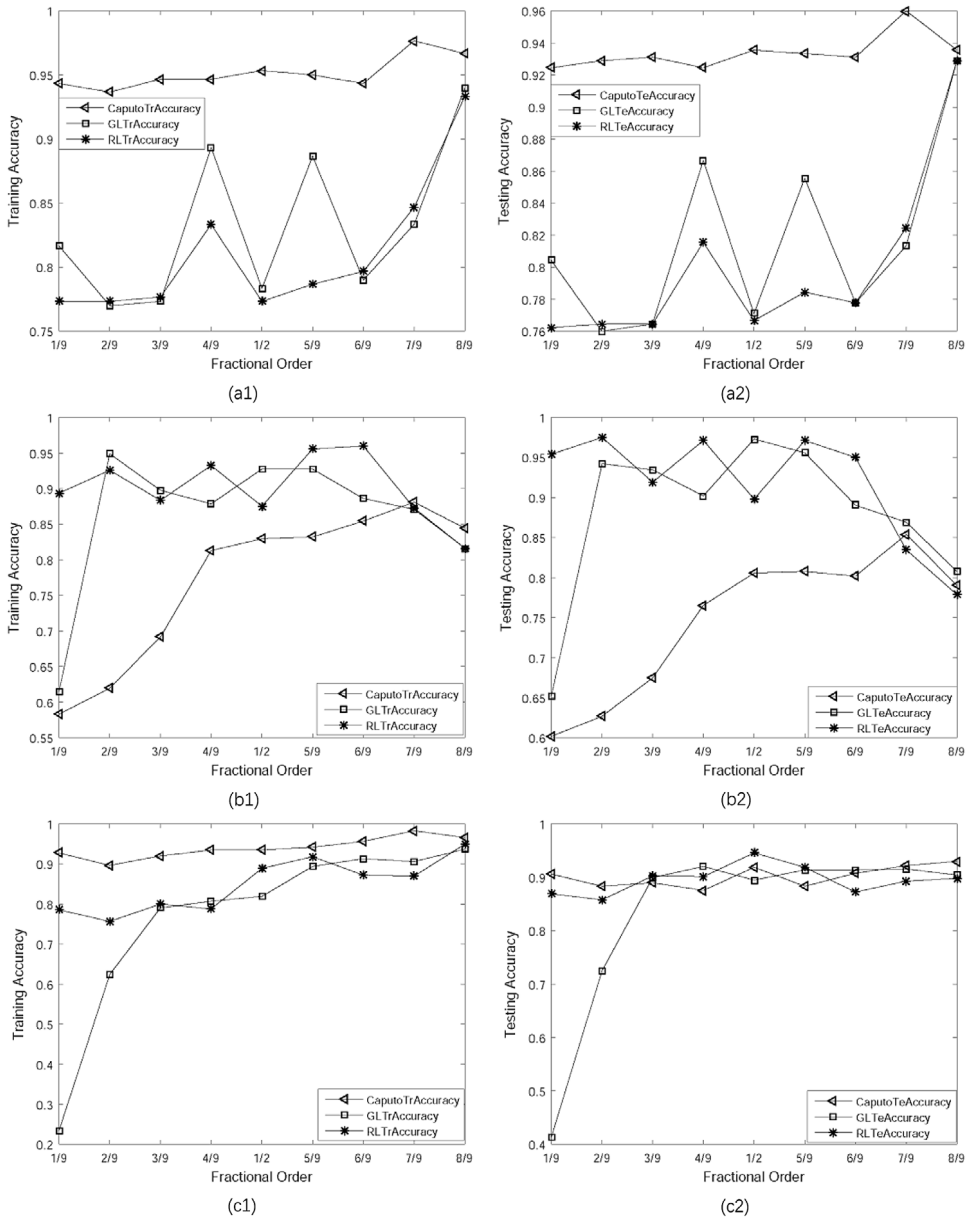

Fig. 1. The performance comparison of different fractional-order BP algorithms in terms of various fractional definitions on datasets: Iris, Liver and Sonar.

Table 3
The comparison of training time on different fractional-order BP algorithms in terms of Iris, Liver and Sonar datasets.

Data sets	Frac defs	1/9	2/9	3/9	4/9	1/2	5/9	6/9	7/9	8/9
1. Iris	Caputo	1.220	1.276	1.257	1.281	1.188	1.245	1.236	1.214	1.279
	GL	1.311	1.276	1.229	1.244	1.149	1.257	1.263	1.296	1.240
	RL	0.628	0.660	0.609	0.606	0.565	0.613	0.659	0.636	0.654
2. Liver	Caputo	6.008	5.883	5.879	5.866	5.362	5.860	5.854	5.866	5.864
	GL	5.853	5.876	5.871	5.860	5.375	5.868	5.887	5.887	5.831
	RL	2.919	2.932	2.934	2.930	2.688	2.936	2.935	2.921	2.928
3. Sonar	Caputo	4.916	4.893	4.893	4.916	4.697	5.021	4.940	4.984	4.893
	GL	4.927	4.909	4.904	4.929	4.708	4.949	5.007	4.936	4.887
	RL	2.449	2.460	2.480	2.475	2.330	2.475	2.491	2.503	2.463

shows us the number of hidden nodes between 200 and 300 is a better choice, since more hidden nodes are more time-consuming computationally. More importantly, we reach an interesting conclusion that $\frac{7}{9}$-order BP algorithms have been observed to have better performances in most cases.

Under these specific parameter settings: learning rate {0.5, 1, 2, 3} and hidden nodes {100, 200, 300, 500}, Figs. 2 and 3 demonstrate that the best fractional-order in training MNIST database should be near $\frac{7}{9}$-order BP algorithm.

To verify the theoretical results of this work, we have redone the simulation (learning rate is 0.5, and 100 hidden nodes) with 1000 training epochs and compare the $\frac{7}{9}$-order and integer-order BP algorithms in terms of the above two performance figures. Fig. 4 shows the training accuracies for up to 1000 training epochs. It

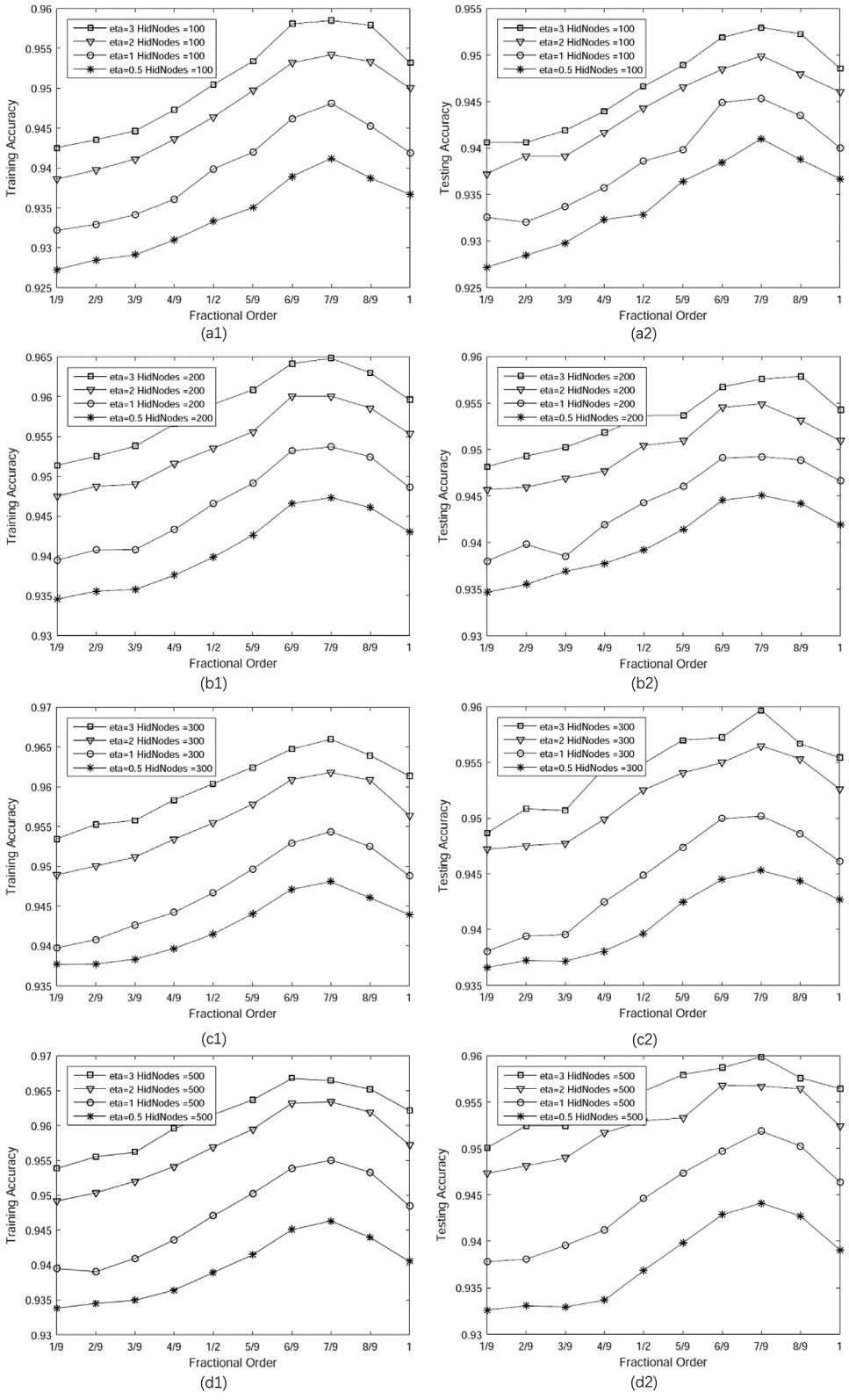

Fig. 2. The comparison of different fractional and integer order BP algorithms for various learning rates with fixed numbers of hidden nodes.

clearly shows that $\frac{7}{9}$-order BP algorithm performs much better than the integer-order BP algorithm. In addition, it performs stable after approximately 400 training epochs. Fig. 5 demonstrates the errors of $\frac{7}{9}$-order BP networks. It is clear to see that the errors are monotonically decreasing. It also shows the norms of gradient of error function with respect to the total weights (25) for $\frac{7}{9}$-order network in Fig. 5. We observe that the $\frac{7}{9}$-order BP algorithms perform stably and tend to converge to zero with increasing iterations. These observations effectively verify the theoretical results of the presented algorithm in this paper, such as monotonicity and convergence.

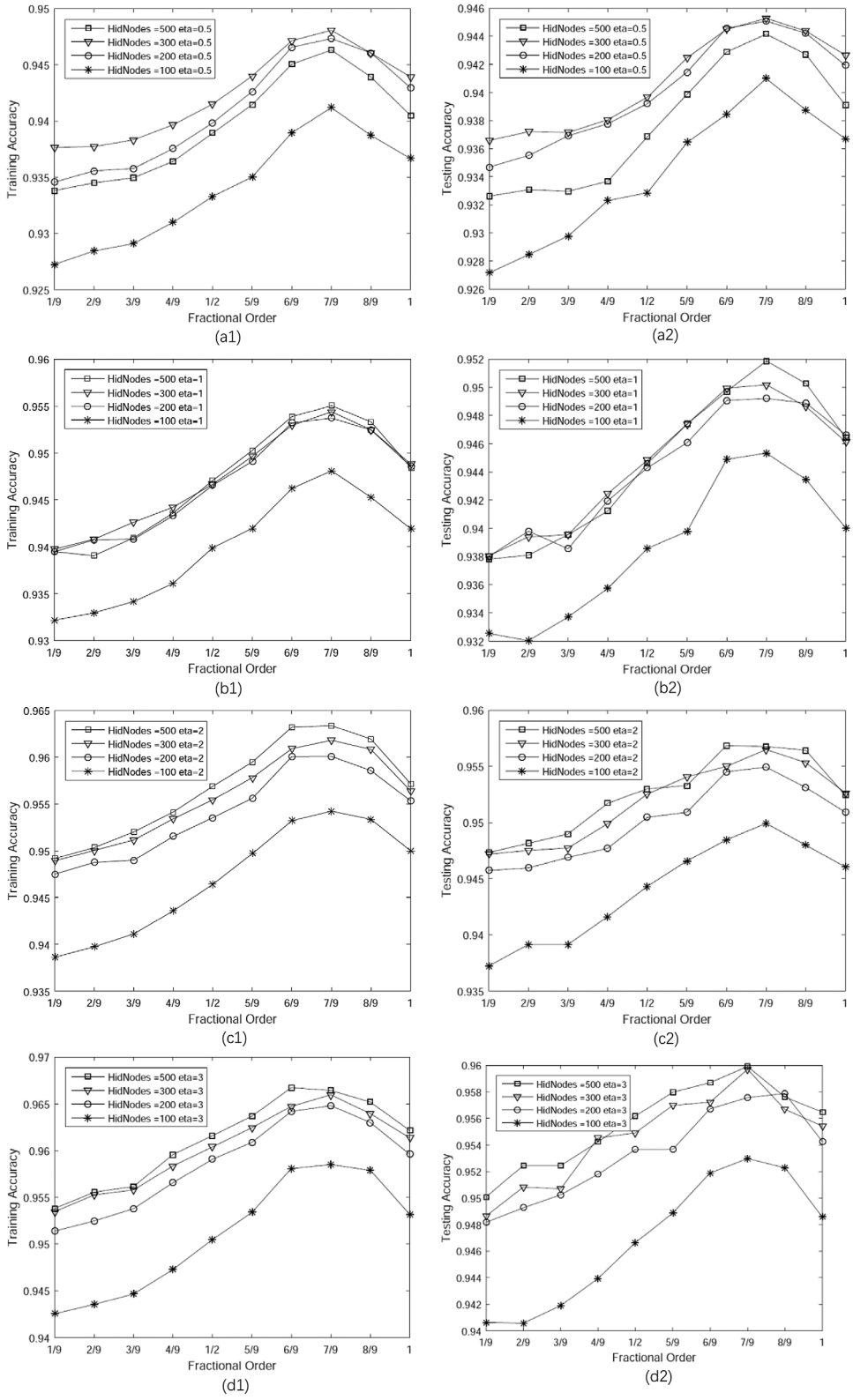

Fig. 3. The comparison of different fractional and integer order BP algorithms for various numbers of hidden nodes at fixed learning rates.

6. Conclusions

In this paper, we have extended the fractional steepest descent approach to BP training of FNNs. The proposed method and its theoretical analyses distinct from the existing results. We have analyzed the convergence of a Caputo fractional-order two-layers neural network trained with BP algorithm. From theoretical point of view, we have proven the monotonicity of error function and weak (strong) convergence of the proposed Caputo fractional-order BP algorithm. The numerical results support the theoretical conclusions very well. Boundedness of weight sequence during training plays an important role in convergent behavior on

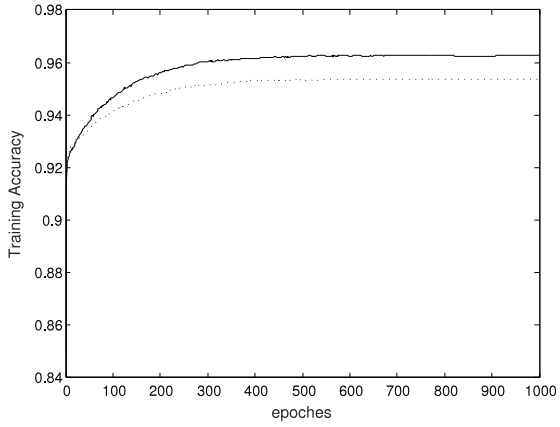

Fig. 4. The training accuracies of $\frac{7}{9}$-order and common integer BP algorithms.

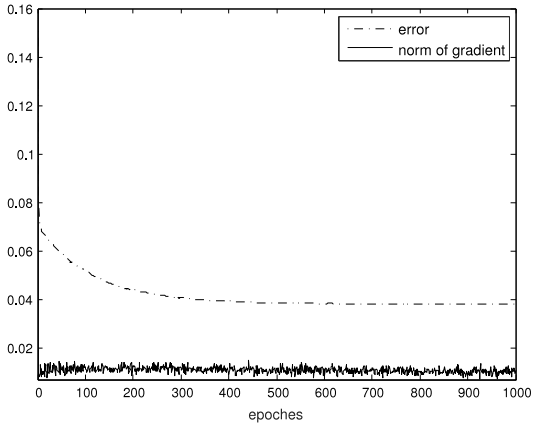

Fig. 5. The training error and the norm of gradient with respect to weights.

both numerical simulations and theoretical analysis in this paper. Our latest numerical simulations demonstrate that the weight sequence by introducing l_2 regularization method approaches to be stable during training. In addition, the trained network improves the generalization much better. Then, how to rigorously prove the boundedness of weight sequence is one attracted topic in our future work.

Acknowledgments

The authors would like to express their gratitude to the editors and anonymous reviewers for their valuable comments and suggestions which improve the quality of this paper.

Appendix

For the sake of description, we introduce the following notations:

$$\Delta u_i^k = u_i^{k+1} - u_i^k = -\eta_c D_{u_i^k}^{\alpha} E(\mathbf{w}); \tag{29}$$

$$\Delta v_{im}^k = v_{im}^{k+1} - v_{im}^k = -\eta_c D_{v_{im}^k}^{\alpha} E(\mathbf{w}); \tag{30}$$

$$\Delta \mathbf{w}^k = \mathbf{w}^{k+1} - \mathbf{w}^k = -\eta_c D_{\mathbf{w}^k}^{\alpha} E(\mathbf{w}); \tag{31}$$

$$G^k = G\left(\mathbf{V}\mathbf{x}^k\right); \tag{32}$$

$$\mathbf{\Psi}^k = G^{k+1} - G^k \tag{33}$$

where $k \in \mathbb{N}$; $i = 1, 2, \ldots, n$; $m = 1, 2, \ldots, p$.

The proof of Theorem 4.1 is divided into four parts dealing with statements (i)–(iv), respectively.

Proof to (i) of Theorem 4.1. $E(\mathbf{w}^{k+1}) \leq E(\mathbf{w}^k),\ k \in \mathbb{N}$.

By the error function (13) we have

$$E(\mathbf{w}^{k+1}) = \sum_{j=1}^{J} f_j(\mathbf{u}^{k+1} \cdot G^{k+1,j}), \tag{34}$$

$$E(\mathbf{w}^k) = \sum_{j=1}^{J} f_j(\mathbf{u}^k \cdot G^{k,j}). \tag{35}$$

By using the Taylor mean value theorem with Lagrange remainder, we have the following estimation

$$\begin{aligned} &E(\mathbf{w}^{k+1}) - E(\mathbf{w}^k) \\ &= \sum_{j=1}^{J} \left(f_j(\mathbf{u}^{k+1} \cdot G^{k+1,j}) - f_j(\mathbf{u}^k \cdot G^{k,j})\right) \\ &= \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})(\Delta \mathbf{u}^k \cdot G^{k,j} + \mathbf{u}^k \cdot \mathbf{\Psi}^{k,j} + \Delta \mathbf{u}^k \cdot \mathbf{\Psi}^{k,j}) \\ &\quad + \frac{1}{2} \sum_{j=1}^{J} f_j''(s_1^{k,j})(\mathbf{u}^{k+1} \cdot G^{k+1,j} - \mathbf{u}^k \cdot G^{k,j})^2 \\ &\triangleq \delta_1 + \delta_2 + \delta_3 + \delta_4, \end{aligned} \tag{36}$$

where $s_1^{k,j} \in \mathbb{R}$ is a constant between $\mathbf{u}^k \cdot G^{k,j}$ and $\mathbf{u}^{k+1} \cdot G^{k+1,j}$, $\delta_1 = \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \Delta \mathbf{u}^k \cdot G^{k,j}$, $\delta_2 = \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \mathbf{u}^k \cdot \mathbf{\Psi}^{k,j}$, $\delta_3 = \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \Delta \mathbf{u}^k \cdot \mathbf{\Psi}^{k,j}$ and $\delta_4 = \frac{1}{2} \sum_{j=1}^{J} f_j''(s_1^{k,j})(\mathbf{u}^{k+1} \cdot G^{k+1,j} - \mathbf{u}^k \cdot G^{k,j})^2$. To study the monotonic property of the given error function, we sequentially analyze the above four items, $\delta_1, \delta_2, \delta_3$ and δ_4. For δ_1, we have that

$$\begin{aligned} \delta_1 &= \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \Delta \mathbf{u}^k \cdot G^{k,j} \\ &= \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \Delta u_1^k g(\mathbf{v}_1^k \cdot \mathbf{x}^j) \\ &\quad + \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \Delta u_2^k g(\mathbf{v}_2^k \cdot \mathbf{x}^j) + \cdots \\ &\quad + \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j}) \Delta u_n^k g(\mathbf{v}_n^k \cdot \mathbf{x}^j). \end{aligned} \tag{37}$$

It is clear to see that this equation is tightly related to Eq. (20). In order to substitute the same parts, the following two cases are considered

Case 1. If $(u_i^k - c)^{1-\alpha} \neq 0$. By (27) and substituting (20) and (29) into (37), we have

$$\begin{aligned} \delta_1 &= -\frac{1}{\eta}(\Delta u_1^k)^2 (1-\alpha)\Gamma(1-\alpha)(u_1^k - c)^{\alpha-1} - \cdots \\ &\quad - \frac{1}{\eta}(\Delta u_n^k)^2 (1-\alpha)\Gamma(1-\alpha)(u_n^k - c)^{\alpha-1} \\ &\leq -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1} \sum_{i=1}^{n} (\Delta u_i^k)^2 \\ &= -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1} \|\Delta \mathbf{u}^k\|^2. \end{aligned} \tag{38}$$

Case 2. If $(u_i^k - c)^{1-\alpha} = 0$. Without loss of generality, we assume that $(u_1^k - c)^{1-\alpha} = 0$. By (20) and (29), it is easy to prove that

${}_cD^{\alpha}_{u_1^k}E(\mathbf{w}) = 0$ and $\Delta u_1^k = 0$ are valid. Hence, (37) can be represented as follows

$$\begin{aligned}\delta_1 &= 0 - \frac{1}{\eta}(\Delta u_2^k)^2(1-\alpha)\Gamma(1-\alpha)(u_2^k - c)^{\alpha-1} - \cdots \\ &\quad - \frac{1}{\eta}(\Delta u_n^k)^2(1-\alpha)\Gamma(1-\alpha)(u_n^k - c)^{\alpha-1} \\ &\le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\sum_{i=1}^{n}(\Delta u_i^k)^2 \\ &= -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\|\Delta\mathbf{u}^k\|^2. \end{aligned} \tag{39}$$

In terms of **Case 1** and **Case 2**, we have the following estimation

$$\delta_1 \le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\|\Delta\mathbf{u}^k\|^2. \tag{40}$$

By using the Taylor mean value theorem with Lagrange remainder, we get

$$\begin{aligned}\delta_2 &= \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\mathbf{u}^k \cdot \boldsymbol{\Psi}^{k,j} \\ &= \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\sum_{i=1}^{n} u_i^k g'(\mathbf{v}_i^k \cdot \mathbf{x}^j)\Delta\mathbf{v}_i^k \cdot \mathbf{x}^j \\ &\quad + \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\frac{1}{2}\sum_{i=1}^{n} u_i^k g''(s_2^{k,j})(\Delta\mathbf{v}_i^k \cdot \mathbf{x}^j)^2 \\ &= \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\sum_{i=1}^{n} u_i^k g'(\mathbf{V}_i^k \cdot \mathbf{x}^j)(\Delta v_{i1}^k x_1^j + \cdots + \Delta v_{ip}^k x_p^j) \\ &\quad + \sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\frac{1}{2}\sum_{i=1}^{n} u_i^k g''(s_2^{k,j})(\Delta\mathbf{v}_i^k \cdot \mathbf{x}^j)^2, \end{aligned} \tag{41}$$

where $s_2^{k,j} \in \mathbb{R}$ is a constant between $\mathbf{v}_i^k \cdot \mathbf{x}^j$ and $\mathbf{v}_i^{k+1} \cdot \mathbf{x}^j$.

It is remarkable to see that the result of above equation is related to Eq. (24). By virtue of the above results, two cases also should be considered in order to substitute the same parts.

Case 3. If $(v_{im}^k - c)^{1-\alpha} \ne 0$. Substituting (24) and (30) into (41) and (27), we have

$$\begin{aligned}\delta_2 &= -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)\sum_{i=1}^{n}(\Delta v_{i1}^k)^2(v_{i1}^k - c)^{\alpha-1} - \cdots \\ &\quad - \frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)\sum_{i=1}^{n}(\Delta v_{ip}^k)^2(v_{ip}^k - c)^{\alpha-1} \\ &\quad + \frac{1}{2}\sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\sum_{i=1}^{n} u_i^k g''(s_2^{k,j})(\Delta\mathbf{v}_i^k \cdot \mathbf{x}^j)^2 \\ &\le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\sum_{i=1}^{n}\sum_{m=1}^{p}(\Delta v_{im}^k)^2 \\ &\quad + \frac{1}{2}\eta J C_1^2 C_3 C_2^2\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2 \\ &\le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2 \\ &\quad + A_1\|\Delta\mathbf{w}^k\|^2, \end{aligned} \tag{42}$$

where $A_1 = \frac{1}{2}\eta J C_1^2 C_3 C_2^2$.

Case 4. If $(v_{im}^k - c)^{1-\alpha} = 0$. Without loss of generality, we assume that $(v_{i1}^k - c)^{1-\alpha} = 0$. Then, by (24) and (30), ${}_cD^{\alpha}_{v_{i1}^k}E(\mathbf{w}^k) = 0$ and $\Delta v_{i1}^k = 0$ are valid. In addition, it holds that

$$\begin{aligned}\delta_2 &= 0 - \frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)\sum_{i=1}^{n}(\Delta v_{i2}^k)^2(v_{i2}^k - c)^{\alpha-1} - \cdots \\ &\quad - \frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)\sum_{i=1}^{n}(\Delta v_{ip}^k)^2(v_{ip}^k - c)^{\alpha-1} \\ &\quad + \frac{1}{2}\sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\sum_{i=1}^{n} u_i^k g''(s_2^{k,j})(\Delta\mathbf{v}_i^k \cdot \mathbf{x}^j)^2 \\ &\le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\sum_{i=1}^{n}\sum_{m=2}^{p}(\Delta v_{im}^k)^2 \\ &\quad + \frac{1}{2}\eta J C_1^2 C_3 C_2^2\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2 \\ &\le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2 \\ &\quad + A_1\|\Delta\mathbf{w}^k\|^2. \end{aligned} \tag{43}$$

According to **Case 3** and **Case 4**, we can obtain that

$$\delta_2 \le -\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2 + A_1\|\Delta\mathbf{w}^k\|^2, \tag{44}$$

where $A_1 = \frac{1}{2}\eta J C_1^2 C_3 C_2^2$.

Based on Cauchy–Schwarz and the fundamental inequalities, it holds the following evaluation

$$\begin{aligned}|\delta_3| &= \left|\sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\Delta\mathbf{u}^k \cdot \boldsymbol{\Psi}^{k,j}\right| \\ &\le \left|\sum_{j=1}^{J} f_j'(\mathbf{u}^k \cdot G^{k,j})\|\Delta\mathbf{u}^k\|\,\|\boldsymbol{\Psi}^{k,j}\|\right| \\ &\le \frac{C_1}{2}\sum_{j=1}^{J}\left(\|\Delta\mathbf{u}^k\|^2 + \|\boldsymbol{\Psi}^{k,j}\|^2\right). \end{aligned} \tag{45}$$

By Lagrange mean-value theorem, we have

$$\begin{aligned}\|\boldsymbol{\Psi}^{k,j}\| &= \|G^{k+1,j} - G^{k,j}\| \\ &= \sqrt{\sum_{i=1}^{n}(g'(s_3^{k,j}))^2(\Delta\mathbf{v}_i^k \cdot \mathbf{x}^j)^2} \\ &\le \sqrt{\sum_{i=1}^{n}(g'(s_3^{k,j}))^2\|\Delta\mathbf{v}_i^k\|^2\|\mathbf{x}^j\|^2} \\ &\le C_1 C_2\sqrt{\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2}, \end{aligned} \tag{46}$$

and

$$\|G^{k,j}\| = \|G(\mathbf{V}^k\mathbf{x}^j)\| \le \sqrt{n}C_1 \triangleq C_4, \tag{47}$$

where $s_3^{k,j} \in \mathbb{R}$ is a constant between $\mathbf{v}_i^k \cdot \mathbf{x}^j$ and $\mathbf{v}_i^{k+1} \cdot \mathbf{x}^j$.

Substituting (46) into (45), we have

$$|\delta_3| \le \frac{C_1}{2}\sum_{j=1}^{J}\left(\|\Delta\mathbf{u}^k\|^2 + C_1^2 C_2^2\sum_{i=1}^{n}\|\Delta\mathbf{v}_i^k\|^2\right)$$

$$\leq \frac{C_1 J}{2}(1 + C_1^2 C_2^2)\|\Delta \mathbf{w}^k\|^2 = A_2\|\Delta \mathbf{w}^k\|^2, \tag{48}$$

where $A_2 = \frac{C_1 J}{2}(1 + C_1^2 C_2^2)$.

Similarly, we have

$$\begin{aligned}|\delta_4| &= \left|\frac{1}{2}\sum_{j=1}^{J} f_j''(s_1^{k,j})(\mathbf{u}^{k+1}\cdot G^{k+1,j} - \mathbf{u}^k \cdot G^{k,j})^2\right| \\ &\leq \frac{C_1}{2}\sum_{j=1}^{J}(\Delta \mathbf{u}^k \cdot G^{k,j} + \mathbf{u}^k \cdot \Psi^{k,j})^2 \\ &\leq \frac{C_1}{2}\sum_{j=1}^{J} 2(\|\Delta \mathbf{u}^k\|^2\|G^{k,j}\|^2 + \|\mathbf{u}^k\|^2\|\Psi^{k,j}\|^2). \end{aligned} \tag{49}$$

Substituting (46) and (47) into (49), we can obtain

$$\begin{aligned}|\delta_4| &\leq \frac{C_1}{2}\sum_{j=1}^{J} 2\left(C_4^2\|\Delta \mathbf{u}^k\|^2 + C_3^2 C_1^2 C_2^2 \sum_{i=1}^{n}\|\Delta \mathbf{v}_i^k\|^2\right) \\ &\leq (JC_1C_4^2 + JC_1^3C_2^2C_3^2)\|\Delta \mathbf{w}^k\|^2 \\ &= A_3\|\Delta \mathbf{w}^k\|^2, \end{aligned} \tag{50}$$

where $A_3 = JC_1C_4^2 + JC_1^3C_2^2C_3^2$.

By assumption (A3), the estimation of (36) can be given as follows

$$\begin{aligned}&E(\mathbf{w}^{k+1}) - E(\mathbf{w}^k) \\ &= \delta_1 + \delta_2 + \delta_3 + \delta_4 \leq \delta_1 + \delta_2 + |\delta_3| + |\delta_4| \\ &\leq \left(-\frac{1}{\eta}(1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1} + A_1 + A_2 + A_3\right) \\ &\quad \times \|\Delta \mathbf{w}^k\|^2 \\ &= \left(-\frac{1}{\eta}A_4 + A_5\right)\|\Delta \mathbf{w}^k\|^2 \leq 0, \end{aligned} \tag{51}$$

where

$$A_4 = (1-\alpha)\Gamma(1-\alpha)(C_3 - c)^{\alpha-1}, \qquad A_5 = A_1 + A_2 + A_3. \tag{52}$$

We note that the upper bound of learning rate, γ, can be determined by Eq. (52), that is, $\gamma = \frac{A_4}{A_5}$.

This then completes the proof of the statement (i) of Theorem 4.1: There exists $E^* \geq 0$ such that $\lim_{k\to\infty} E(\mathbf{w}^k) = E^*$.

Proof to (ii) of Theorem 4.1. From the conclusion of (i), we know that $E(\mathbf{w}^{k+1}) \leq E(\mathbf{w}^k)$, $E(\mathbf{w}^k) \geq 0$ and $E(\mathbf{w}^k)$ is also bounded below. Hence there exists $E^* \geq 0$ such that

$$\lim_{k\to\infty} E(\mathbf{w}^k) = E^*. \tag{53}$$

The proof of (ii) is thus completed.

Proof to (iii) of Theorem 4.1. $\lim_{k\to\infty} \|_c D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^k)\| = 0$.

Based on the result of (51), we can obtain that

$$E(\mathbf{w}^{k+1}) - E(\mathbf{w}^k) \leq \left(-\frac{1}{\eta}A_4 + A_5\right)\|\Delta \mathbf{w}^k\|^2 \leq 0. \tag{54}$$

Let $\beta = \frac{1}{\eta}A_4 - A_5$, we have the following formula

$$\begin{aligned}E(\mathbf{w}^{k+1}) &\leq E(\mathbf{w}^k) - \beta\|\Delta \mathbf{w}^k\|^2 \\ &\leq E(\mathbf{w}^{k-1}) - \beta(\|\Delta \mathbf{w}^{k-1}\|^2 + \|\Delta \mathbf{w}^k\|^2) \\ &\leq E(\mathbf{w}^0) - \beta\sum_{l=0}^{k}\|\Delta \mathbf{w}^l\|^2. \end{aligned} \tag{55}$$

Since $E(\mathbf{w}^{k+1}) \geq 0$, we then get that

$$\beta\sum_{l=0}^{k}\|\Delta \mathbf{w}^l\|^2 \leq E(\mathbf{w}^0). \tag{56}$$

Let $k \to \infty$, it holds that

$$\sum_{l=0}^{\infty}\|\Delta \mathbf{w}^l\|^2 \leq \frac{1}{\beta}E(\mathbf{w}^0) < \infty. \tag{57}$$

By the rule of series convergence, the general term has the following result

$$\lim_{k\to\infty}\|\Delta \mathbf{w}^k\| = 0. \tag{58}$$

According to (31), it is easy to obtain that

$$\lim_{k\to\infty}\|D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^k)\| = 0. \tag{59}$$

The proof is thus completed.

Proof to (iv) of Theorem 4.1. We note that the error function $E(\mathbf{w})$ is continuous and differentiable on the compact set $\Phi \subset \mathbb{R}^{n(p+1)}$, furthermore, the weight sequence satisfies that

$$\lim_{k\to\infty}\|D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^k)\| = 0, \qquad \lim_{k\to\infty}\|\Delta \mathbf{w}^k\| = 0. \tag{60}$$

Due to the fact that the fractional derivatives (20) and (24) are both continuous on Φ, thus the corresponding gradient $D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^k)$ is also continuous. The weight sequence $\{\mathbf{w}^k\}$ ($k \in \mathbb{N}$) has a subsequence $\{\mathbf{w}^{k_i}\}$ ($i \in \mathbb{N}$) which is convergent to, say, $\mathbf{w}^*$. Then, $\lim_{i\to\infty} \mathbf{w}^{k_i} = \mathbf{w}^*$. According to the continuity of $D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^k)$, we can obtain that

$$\lim_{i\to\infty} D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^{k_i}) = D_{\mathbf{w}}^{\alpha} E(\mathbf{w}^*) = 0. \tag{61}$$

By the assumption (A4), $\widehat{\Phi}$ is the α-order stationary point set of the error function $E(\mathbf{w})$. Obviously, the limit point $\mathbf{w}^* \in \widehat{\Phi}$. We can induce that any limit point is a stationary point of E.

Let $\widetilde{\Phi}$ be the set of limit points $\{\mathbf{w}^k\}$. Then, it holds that $\widetilde{\Phi} \subset \widehat{\Phi}$ based on the above discussion. Suppose that the set of limit points is $\widetilde{\Phi} = \{z_1, z_2, \ldots, z_m\}(m > 1)$, $\delta = \min\{\|z_i - z_j\| \mid i \neq j, i, j = 1, \ldots, m\} > 0$, and that $N(z_i, \delta) = \{s \mid \|\mathbf{w} - z_i\| \leq \delta\}$. We can choose an integer $k_0 \geq 0$ such that $\mathbf{w}^k \in \cup_{i=1}^{m} N(z_i, \frac{\delta}{4})$ and $\|\mathbf{w}^k - \mathbf{w}^{k+1}\| \leq \frac{\delta}{4}$ for all $k \geq k_0$. There exists $k_1 \geq k_0$ such that $\mathbf{w}^{k_1} \in N(z_1, \frac{\delta}{4})$. In addition, we have

$$\begin{aligned}\|z_i - \mathbf{w}^{k_1+1}\| &\geq \|z_i - z_1\| - (\|z_1 - \mathbf{w}^{k_1}\| + \|\mathbf{w}^{k_1} - \mathbf{w}^{k_1+1}\|) \\ &\geq \delta - \frac{2\delta}{4} = \frac{\delta}{2}, \quad (i \geq 2). \end{aligned} \tag{62}$$

It shows that $\mathbf{w}^{k_1+1}$ is not in the neighborhood $N(z_i, \frac{\delta}{4})$ for $i \geq 2$. Then, we have that $\mathbf{w}^{k_1+1} \in N(z_1, \frac{\delta}{4})$. By mathematical induction, it can be obtained that $\mathbf{w}^k \in N(z_1, \frac{\delta}{4})$ for all $k \geq k_1$. However, this is contrast with the condition (A4) which states that there are a finite different number stationary points. Then, we have $m = 1$ and this completes the proof of (iv) of Theorem 4.1.

References

Chen, X. (2013). *Application of fractional calculus in bp neural networks*. (Ph.D. thesis), Nanjing, Jiangsu: Nanjing Forestry University.

Chen, B., & Chen, J. (2016). Global $o(t^{-\alpha})$ stability and global asymptotical periodicity for a non-autonomous fractional-order neural networks with time-varying delays. *Neural Networks*, *73*, 47–57.

Delavari, H., Baleanu, D., & Sadati, J. (2012). Stability analysis of caputo fractional-order nonlinear systems revisited. *Nonlinear Dynamics*, *67*(4), 2433–2439.

Deng, W., & Li, C. (2005). Chaos synchronization of the fractional lü system. *Physica A. Statistical Mechanics and its Applications*, *353*, 61–72.
Gorenflo, R., & Mainardi, F. (2008). Fractional calculus: integral and differential equations of fractional order. *Mathematics*, *49*(2), 277–290.
Kvitsinskii, A. A. (1993). Fractional integrals and derivatives: theory and applications. *Theoretical and Mathematical Physics*, *3*, 397–414.
Love, E. R. (1971). Fractional derivatives of imaginary order. *Journal of the London Mathematical Society*, *3*(2), 241–259.
McBride, A. C. (1986). *Fractional calculus*. USA: Halsted.
Miller, K. S. (1995). Derivatives of noninteger order. *Mathematics Magazine*, *68*, 183–192.
Nielsen, M. 2016. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/chap1.html.
Nishimoto, K. (1989). *Fractional calculus: integrations and differentiations of arbitrary order*. New Haven, CT, USA: New Haven Univ. Press.
Oldham, K. B., & Spanier, J. (1974). *The fractional calculus: theory and applications of differentiation and integration to arbitrary order*. USA: Academic.
Pu, Y., Zhou, J., Zhang, Y., Zhang, N., Huang, G., & Siarry, P. (2015). Fractional extreme value adaptive training method: fractional steepest descent approach. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(4), 653–662.
Rakkiyappan, R., Cao, J., & Velmurugan, G. (2015). Existence and uniform stability analysis of fractional-order complex-valued neural networks with time delays. *IEEE Transaction on Neural Networks and Learning Systems*, *26*(1), 84–97.
Rakkiyappan, R., Sivaranjani, R., Velmurugan, G., & Cao, J. (2016). Analysis of global $o(t^{-\alpha})$ stability and global asymptotical periodicity for a class of fractional-order complex-valued neural networks with time varying delays. *Neural Networks*, *77*, 51–69.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back propagating errors. *Nature*, *323*(9), 533–536.
Shao, H., Wu, W., & Liu, L. (2010). Convergence of online gradient method with penalty for bp neural networks. *Communications in Mathematical Research*, *26*(1), 67–75.
Wang, H., Yu, Y., & Wen, G. (2014). Stability analysis of fractional-order hopfield neural networks with time delays. *Neural Networks*, *55*, 98–109.
Wang, H., Yu, Y., Wen, G., Zhang, S., & Yu, J. (2015). Global stability analysis of fractional-order hopfield neural networks with time delay. *Neurocomputing*, *154*, 15–23.
Wu, X., Li, J., & Chen, G. (2008). Chaos in the fractional order unified system and its synchronization. *Journal of the Franklin Institute*, *345*(4), 392–401.
Wu, W., Shao, H., & Li, Z. (2006). Convergence of batch bp algorithm with penalty for fnn training. In *Neural information processing, Vol. 4232* (pp. 562–569).
Wu, W., Wang, J., Cheng, M., & Li, Z. (2011b). Convergence analysis of online gradient method for bp neural networks. *Neural Networks*, *24*, 91–98.
Wu, A., & Zen, Z. (2013). Anti-synchronization control of a class of memristive recurrent neural networks. *Commnications in Nonlinear Science and Numerical Simulation*, *18*(2), 373–385.
Wu, A., Zhang, J., & Zen, Z. (2011a). Dynamic behaviors of a class of memristor-based hopfield networks. *Physics Letters A*, *375*(15), 1661–1665.
Xiao, M., Zheng, W., Jiang, G., & Cao, J. (2015). Undamped oscillations generated by hopf bifurcations in fractional-order recurrent neural networks with caputo derivative. *IEEE Transaction on Neural Networks and Learning Systems*, *26*(12), 3201–3214.
Xu, Z., Zhang, R., & Jin, W. (2009). When does online bp training converge? *IEEE Transactions on Neural Networks*, *20*(10), 1529–1539.
Zhang, S., Yu, Y., & Wang, H. (2015). Mittag-leffler stability of fractional-order hopfield neural networks. *Nonlinear Analysis. Hybrid Systems*, *16*, 104–121.