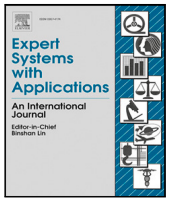
Expert Systems With Applications 236 (2024) 121166

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Adaptive stepsize estimation based accelerated gradient descent algorithm for fully complex-valued neural networks

Weijing Zhao, He Huang *

School of Electronics and Information Engineering, Soochow University, Suzhou 215006, PR China

ARTICLE INFO

Keywords:
Accelerated gradient descent
Adaptive stepsize
Local smoothness constant
Curvature information
Fully complex-valued neural networks

ABSTRACT

Nesterov accelerated gradient (NAG) method is an efficient first-order algorithm for optimization problems. To ensure the convergence, it usually takes a relatively conservative constant as the stepsize. However, the choice of stepsize has a great impact on the optimization process. In this paper, two adaptive stepsize estimation methods are proposed for complex-valued NAG algorithm for efficient training of fully complex-valued neural networks. The basic idea of the first one is to adaptively determine suitable stepsize by estimating the local smoothness constant of the loss function with the norm of approximate complex Hessian matrix. Its validity is theoretically analyzed by means of the decomposition of complex matrix. Furthermore, by introducing a new parameter design method for multi-step quasi-Newton condition, an improved stepsize estimation is presented. Experimental results on pattern recognition, channel equalization, wind forecasting and synthetic aperture radar (SAR) target classification demonstrate the effectiveness of the proposed methods.

1. Introduction

In recent years, complex-valued neural networks (CVNNs) have been widely applied in various engineering fields for that they have been proved superior to real-valued neural networks (RVNNs) in many areas, including wireless communication, synthetic aperture radar (SAR) and medical image denoising, etc. (Chen et al., 2022; Peker, 2021; Rawat et al., 2021; Scarnati & Lewis, 2021; Singhal et al., 2022; Xie et al., 2020; Zeng et al., 2022). Actually, compared with RVNNs, CVNNs are naturally suitable for processing complex-valued data, because the relation between the real and imaginary parts (or the magnitude and phase) is efficiently considered. In addition, CVNNs also have the advantages of fast learning and strong generalization ability (Aizenberg, 2011; Ceylan et al., 2011; Nitta, 2004).

According to the activation functions, CVNNs are divided into two categories: split CVNNs and fully CVNNs (Zhang et al., 2019). The difference between them is that, in split CVNNs, complex-valued input is split into the real-imaginary or magnitude-phase form by activation functions. While fully CVNNs process complex-valued input directly. One of the advantages of fully CVNNs is that the inherent correlation within complex-valued data can be well characterized (Sorber et al., 2012) and thus better performance is usually guaranteed.

As one of the most popular training algorithms of RVNNs, the back-propagation (BP) algorithm was extended to the complex domain as split complex BP algorithms (Leung & Haykin, 1991; Zhang et al., 2009) and fully complex BP algorithms (Li & Adalı, 2008; Zhang et al., 2014). However, similar to their counterparts in real domain, these complex-valued gradient descent (CGD) methods suffer from slow convergence, local minima and saddle points (Nitta, 2013).

Actually, in real domain, many accelerated strategies have been proposed to resolve the disadvantages of gradient descent algorithms (Rojc & Mlakar, 2020). One of them is the famous Nesterov accelerated gradient (NAG) method (Nesterov, 1983), which is a modification of the heavy-ball method (Polyak, 1964) and has been proved to have the convergence rate $O(1/t^2)$ for solving convex problems. In Zhang and Xia (2015), NAG was extended to the complex domain as complex-valued NAG (CNAG) method to deal with quadratic programming problems. It is noted that, in these works, both NAG and CNAG usually take fixed stepsize during the iteration process. To guarantee the convergence of the two algorithms, the objective function to be optimized is supposed to be convex and smooth. Meanwhile, when the objective function is L-smooth and its global smoothness constant is available, the stepsize is suggested to be set as $\frac{1}{L}$, by which the convergence of NAG or CNAG can be guaranteed (Nesterov, 1983; Zhang & Xia, 2015). However, for the objective function of neural networks, its global smoothness constant is generally hard to be calculated, since it is an NP-hard problem (Virmaux & Scaman, 2018). Worse yet, due to the existence of singularities in activation functions of fully CVNNs, the global smoothness constant of the objective function may not exist. Under these circumstances, when CNAG is applied for the training of fully CVNNs, it is not easy to choose suitable stepsize to

* Corresponding author.
E-mail addresses: zhao_wj_edu@163.com (W. Zhao), hhuang@suda.edu.cn (H. Huang).

https://doi.org/10.1016/j.eswa.2023.121166
Received 23 December 2022; Received in revised form 30 June 2023; Accepted 8 August 2023
Available online 23 August 2023

achieve satisfactory performance. Therefore, it is of great significance to investigate the stepsize design problem of CNAG.

Due to the disadvantages of fixed stepsize, some adaptive stepsize methods for NAG in real domain were recently proposed, such as Nesterov accelerated adaptive moment estimation (Nadam) (Dozat, 2016), in which Nesterov momentum was incorporated into Adam. On the other hand, the back-tracking (or line search) method can be also employed to find the stepsize (Beck, 2017). However, at each iteration, line search may cost expensively to obtain a suitable stepsize. Recently, some methods were presented to estimate the local smoothness constant of the objective function in real domain, which is beneficial for the design of stepsize (Malitsky & Mishchenko, 2020; Nishioka & Kanno, 2021). In Malitsky and Mishchenko (2020), a rough estimation of local smoothness constant was introduced. There is still room for improvement. Furthermore, to our knowledge, this kind of issue has not yet been studied for CNAG based training algorithm of fully CVNNs.

Motivated by the above discussion, in this paper, an adaptive stepsize estimation method is proposed for CNAG (which is termed as AS-CNAG for simplicity) and applied for efficient training of fully CVNNs. The basic idea of AS-CNAG is that the stepsize is determined by using the estimated local smoothness constant. Inspired by complex Barzilai–Borwein method (CBBM) (Zhang & Mandic, 2015), an estimation of local smoothness constant of the loss function can be easily obtained by means of approximate second-order information. Based on the properties of complex matrix, theoretical analysis is presented to support the feasibility of the design method. It is noted that, for the CNAG method in Zhang and Xia (2015), the stepsize is determined by line search, which is time-consuming for the training of CVNNs. However, in our algorithm, the stepsize is adaptively obtained by estimating the local smoothness constant of the loss function to improve the training efficiency of fully CVNNs.

Furthermore, it is noted that traditional CBBM built on the secant equation (i.e., quasi-Newton condition) only uses the information involved in the most recent two iterations. Its performance would be improved by multi-step quasi-Newton condition (Ford & Moghrabi, 1994). In the multi-step quasi-Newton method, iteration points are fitted by a curve such that the information obtained in previous iterations can be fully utilized by polynomial interpolation. One way to calculate the parameters of the curve is the unit-spaced method (Ford & Moghrabi, 1994). However, this method may result in the unstable phenomenon during the iteration process. To solve this problem, some methods try to find the parameters according to the distance between different iteration points, such as the accumulative and fixed-point approaches (Ford & Moghrabi, 1993). Then, to obtain more appropriate parameters of the interpolation, some kinds of implicit methods were designed (Ford, 2001; Moghrabi, 2017; Moughrabi, 2009). In this paper, by taking multi-step quasi-Newton condition (for simplicity, only two-step case is considered here) into consideration, an improved version of AS-CNAG is further presented for the stepsize estimation and is named as AS-CNAG-multi. It is emphasized that, in our approach, the calculation of the parameters of the interpolation curve is different from the ones mentioned above. Instead, we directly obtain them by measuring the variation of the curvature information between successive iterations. It is expected to give a better estimation for local smoothness constant. Compared with the algorithms proposed in Ford and Moghrabi (1993), our method makes full use of the information involved in the curvature vector pairs and line search is no longer needed to reduce the computational complexity.

Finally, experimental results on pattern classification, nonlinear channel equalization, wind forecasting and SAR target classification are given to illustrate the effectiveness and advantages of AS-CNAG and AS-CNAG-multi over some existing algorithms. The contributions of this study are as follows: (i) To accelerate the convergence and avoid manual tuning, two adaptive methods are proposed for the determination of stepsize of CNAG based on the estimated local smoothness constant of the training loss with the norm of approximate complex Hessian matrix. (ii) Theoretical analysis is presented to confirm the validity of the proposed stepsize design method. (iii) By measuring the importance of curvature information among successive iterations, an adaptive strategy is designed for the calculation of the scaling parameter of multi-step quasi-Newton condition. It provides a better estimation for local smoothness constant and thus more suitable stepsize is obtained. (iv) Demonstrated by experimental results, our methods guarantee faster convergence and better performance for the training of fully CVNNs.

2. Preliminaries

2.1. Fully CVNNs

We consider a feedforward fully CVNN with L layers. Let the activation function of the lth layer be $f_l(\cdot)$, the output of the lth layer can be expressed by

$$\mathbf{h}_l = f_l(\mathbf{w}_l\mathbf{h}_{l-1} + \mathbf{b}_l), \tag{1}$$

where $l = 2, 3, \ldots, L$, $\mathbf{h}_{l-1}$ is the output of the $(l-1)$th layer, $\mathbf{w}_l$ and $\mathbf{b}_l$ are the complex-valued weight matrix between the lth and $(l-1)$th layers and the bias vector of the neurons in the lth layer, respectively.

Assume that the training sample pairs are given by $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{J} \subset \mathbb{C}^M \times \mathbb{C}^N$, where $\mathbf{x}_j$ is the input vector and $\mathbf{y}_j$ is the desired output. The loss function for the training of this kind of fully CVNNs is defined by

$$J(\mathbf{w}) = \sum_{j=1}^{J} (\mathbf{o}_j - \mathbf{y}_j)^H (\mathbf{o}_j - \mathbf{y}_j), \tag{2}$$

where $(\cdot)^H$ represents the conjugate transpose, $\mathbf{w}$ denotes all the adjustable parameters of the fully CVNN, and $\mathbf{o}_j$ is the actual final output corresponding to the jth sample.

2.2. Wirtinger calculus

Consider a function $f(z) : \mathbb{C} \to \mathbb{C}$, which is described by

$$f(z) = u(x, y) + iv(x, y), \tag{3}$$

where x and y are respectively the real and imaginary parts of z, $i = \sqrt{-1}$, both $u(x, y)$ and $v(x, y)$ are real functions of x and y.

For fully CVNNs, it is known from (2) that the loss function is a real function with complex variables. According to Cauchy–Riemann conditions, it is nonanalytic. In this situation, Wirtinger calculus (Kreutz-Delgado, 2009; Wirtinger, 1927) is applicable. In fact, $f(z)$ can be transferred as a function of z and its conjugate z^* since z and z^* can be represented by x and y. As a result, $\partial f/\partial z$ and $\partial f/\partial z^*$ are well defined and can be computed independently. For that $x = \frac{z+z^*}{2}$ and $y = \frac{z-z^*}{2i}$, by the chain rule, one can obtain

$$\begin{aligned} \frac{\partial f}{\partial z} &= \frac{1}{2}\left(\frac{\partial f}{\partial x} - \frac{\partial f}{\partial y} i\right), \\ \frac{\partial f}{\partial z^*} &= \frac{1}{2}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} i\right). \end{aligned} \tag{4}$$

When $f(z)$ is a real function (i.e., $v(x, y) = 0$), it is easy to prove the following property

$$\left(\frac{\partial f}{\partial z}\right)^* = \frac{\partial f}{\partial z^*}. \tag{5}$$

In sequel, the complex gradient $\frac{\partial f}{\partial z^*}$ is denoted by $\nabla_{z^*} f(z)$.

2.3. NAG

In Nesterov (1983), Nesterov proposed a simple but effective accelerated gradient descent method. It was proved that NAG has a convergence rate of $O(1/t^2)$ in the convex setting, where t is the iteration number. For an optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$ in real domain,

let $\mathbf{x}_0 = \mathbf{y}_0$ be an arbitrary initial point, the formula of NAG is described as the following two steps:

$$\begin{aligned}\mathbf{y}_{t+1} &= \mathbf{x}_t - \alpha\nabla f(\mathbf{x}_t),\\ \mathbf{x}_{t+1} &= \mathbf{y}_{t+1} + \beta(\mathbf{y}_t - \mathbf{y}_{t+1}),\end{aligned} \tag{6}$$

with $\beta < 0$.

When f is convex and L-smooth, it can be proved that, for $\alpha = \frac{1}{L}$,

$$f(\mathbf{y}_t) - f(\mathbf{x}^*) \le \frac{2L\left\|\mathbf{x}_1 - \mathbf{x}^*\right\|^2}{t^2}. \tag{7}$$

By introducing $\mathbf{v}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$ with $\mathbf{v}_0 = 0$, (6) can be equivalently rewritten as the most commonly used form (Yang et al., 2016):

$$\begin{aligned}\mathbf{v}_{t+1} &= \beta\mathbf{v}_t - \alpha\nabla f(\mathbf{y}_t + \beta\mathbf{v}_t),\\ \mathbf{y}_{t+1} &= \mathbf{y}_t + \mathbf{v}_{t+1}.\end{aligned} \tag{8}$$

In Zhang and Xia (2015), the complex-valued form of NAG (i.e., CNAG) was presented as a fast complex-valued optimization algorithm to deal with complex quadratic programming problems. The convergence of CNAG is expressed by

$$f(\mathbf{z}_t) - f(\mathbf{z}^*) \le \frac{4\widetilde{L}\left\|\mathbf{z}_1 - \mathbf{z}^*\right\|^2}{t^2}, \tag{9}$$

where $\widetilde{L}$ is the smoothness constant of f with complex variables and the sequence $\{\mathbf{z}_t\}$ is generated by CNAG.

2.4. Multi-step Quasi-Newton method

The multi-step quasi-Newton method proposed by Ford and Moghrabi (1994) is a generalized version of the quasi-Newton one, which uses the information of the latest m iterations by polynomial interpolation. The multi-step quasi-Newton condition is expressed by

$$\mathbf{B}_{t+1}\mathbf{r}_t = \boldsymbol{\omega}_t, \tag{10}$$

where $\mathbf{r}_t$ and $\boldsymbol{\omega}_t$ are represented by the most recent vectors $\{\mathbf{s}_{t-j}\}_{j=0}^{m-1}$ and $\{\mathbf{y}_{t-j}\}_{j=0}^{m-1}$ with $\mathbf{s}_{t-j} = \mathbf{x}_{t-j} - \mathbf{x}_{t-j-1}$ and $\mathbf{y}_{t-j} = \nabla f(\mathbf{x}_{t-j}) - \nabla f(\mathbf{x}_{t-j-1})$. When $m = 2$, the update conditions are described by

$$\begin{aligned}\mathbf{r}_t &= \mathbf{s}_t - \frac{\delta^2}{(2\delta+1)}\mathbf{s}_{t-1},\\ \boldsymbol{\omega}_t &= \mathbf{y}_t - \frac{\delta^2}{(2\delta+1)}\mathbf{y}_{t-1},\end{aligned} \tag{11}$$

where $\delta = \frac{(\tau_2 - \tau_1)}{(\tau_1 - \tau_0)}$.

There are many different approaches to defining $\{\tau_j\}_{j=0}^{m}$, such as the unit-spaced, accumulative and fixed point methods (Ford & Moghrabi, 1993). It is known that the performance of multi-step quasi-Newton method is usually better than the one of standard quasi-Newton method. This is because a better approximation of Hessian matrix can be obtained by using more curvature information.

3. Adaptive stepsize estimation

3.1. AS-CNAG

The original NAG (6) usually adopts fixed stepsize. To ensure convergence and facilitate analysis, the stepsize is suggested to be $\frac{1}{L}$. On the other hand, it is known from (7) or (9) that, the convergence rate of NAG or CNAG is closely related to the global smoothness constant L. It means that the smoothness constant L is an important parameter for NAG or CNAG. However, in practical applications, it is hard to exactly calculate the global smoothness constant of the objective function, especially for fully CVNNs. Therefore, one common way to determine the stepsize is to take a relatively conservative constant to ensure the convergence. This would undoubtedly degrade the convergence speed.

Note that the global smoothness constant is hard (or even impossible) to be obtained, some methods try to determine the stepsize by estimating local smoothness constant, such as AdGD-accel (Malitsky & Mishchenko, 2020). However, there is a lack of research in complex domain.

Recall the definition of local Lipschitz constant in real domain (Marsden & Hoffman, 1993) and note that the loss function of fully CVNNs is a real function of complex variables, the definition of local smoothness constant of this kind of complex functions is given as follows.

Definition 1. A function $f : A \subset \mathbb{C}^n \to \mathbb{R}$ is local smooth at $\mathbf{z}_0 \in A$ if there exist constants $\delta > 0$ and $\hat{L} > 0$ such that

$$\|\nabla_{\mathbf{z}^*} f(\mathbf{z}) - \nabla_{\mathbf{z}^*} f(\mathbf{z}_0)\| \le \hat{L}\|\mathbf{z} - \mathbf{z}_0\| \tag{12}$$

holds for any $\mathbf{z}$ satisfying $\|\mathbf{z} - \mathbf{z}_0\| < \delta$, where $\hat{L}$ is the local smoothness constant.

In AdGD-accel, $\frac{1}{2\hat{L}}$ is estimated in terms of $\frac{\|\mathbf{x}_t - \mathbf{x}_{t-1}\|}{2\|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})\|}$ (Malitsky & Mishchenko, 2020). In other words, $\hat{L}$ is estimated by $\frac{\|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})\|}{\|\mathbf{x}_t - \mathbf{x}_{t-1}\|}$, which is a very rough estimation to $\hat{L}$. In this paper, an adaptive stepsize design method is proposed for CNAG, in which the local smoothness constant is estimated with second-order information. It gives a more accurate approximation compared to AdGD-accel.

In complex domain, CNAG is expressed by

$$\begin{aligned}\mathbf{v}_t &= \beta\mathbf{v}_{t-1} - \alpha\nabla_{\mathbf{w}^*} J(\mathbf{w}_t + \beta\mathbf{v}_{t-1}),\\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \mathbf{v}_t.\end{aligned} \tag{13}$$

It can be also represented by

$$\mathbf{w}_{t+1} = \hat{\mathbf{w}}_{t+1} - \alpha\nabla_{\mathbf{w}^*} J(\hat{\mathbf{w}}_{t+1}), \tag{14}$$

where $\hat{\mathbf{w}}_{t+1} = \mathbf{w}_t + \beta\mathbf{v}_{t-1}$ is the look-ahead position of $\mathbf{w}_t$.

Note that, according to Definition 1, local smoothness can be defined at a point. Therefore, without loss of generality, it is assumed that the loss function (2) is local smooth at the iteration points generated by our method (i.e., the iteration points are not singular points of the loss function). By taking two adjacent points $\hat{\mathbf{w}}_{t+1}$ and $\hat{\mathbf{w}}_t$, it is known that the approximation of complex Hessian matrix satisfies

$$\mathbf{B}_{t+1}\mathbf{s}_t = \mathbf{y}_t, \tag{15}$$

where $\mathbf{s}_t = \hat{\mathbf{w}}_{t+1} - \hat{\mathbf{w}}_t, \mathbf{y}_t = \nabla_{\mathbf{w}^*} f(\hat{\mathbf{w}}_{t+1}) - \nabla_{\mathbf{w}^*} f(\hat{\mathbf{w}}_t)$.

Obviously, one has

$$\|\mathbf{y}_t\| = \|\mathbf{B}_{t+1}\mathbf{s}_t\| \le \|\mathbf{B}_{t+1}\|\|\mathbf{s}_t\|. \tag{16}$$

Therefore, we can use $\|\mathbf{B}_{t+1}\|$ as an estimation of the local smoothness constant $\hat{L}$ in Definition 1. Actually, $\|\mathbf{B}_{t+1}\|$ can be used to guide the design of stepsize for that it provides the measure of the curvature. When $\|\mathbf{B}_{t+1}\|$ is large, it means the function changes rapidly at the current point and so the stepsize should be small, which can help prevent overshooting the minimum. While, if the function has a small curvature, the stepsize would be large, which can help the algorithm converge more quickly.

One efficient way to calculate $\|\mathbf{B}_{t+1}\|$ is Newton or Quasi-Newton method. However, these second-order methods usually require considerable computation and storage costs. Note that CNAG is a first-order method, it is practical that the calculation of $\|\mathbf{B}_{t+1}\|$ should be as simple as possible. Thanks to the work in Barzilai and Borwein (1988), the so-called Barzilai–Borwein method (BBM) was proposed, which was generalized as CBBM in Zhang and Mandic (2015). The basic idea of this method is to replace $\mathbf{B}_{t+1}$ with $\lambda_t\mathbf{I}$, where λ_t is the solution of the least squares problem and can be calculated by

$$\lambda_{1,t} = \frac{\mathbf{s}_t^H\mathbf{y}_t}{\mathbf{s}_t^H\mathbf{s}_t}, \tag{17}$$

or

$$\lambda_{2,t} = \frac{\mathbf{y}_t^H\mathbf{s}_t}{\mathbf{y}_t^H\mathbf{y}_t}. \tag{18}$$

Actually, at the tth iteration, $|\lambda_2|$ is an approximation of the 2-norm of $\mathbf{B}$ to some extent. Here, for simplicity, the subscript t is omitted. Let the quasi-Newton matrix $\mathbf{B} \in \mathbb{C}^{n\times n}$ be Hermitian and positive definite (Sorber et al., 2012). It can be singularly decomposed into the following formula (according to Theorem 2.5.6 in Horn and Johnson (2012))

$$\mathbf{B} = \mathbf{Q}\mathbf{D}\mathbf{Q}^H, \tag{19}$$

where $\mathbf{Q} \in \mathbb{C}^{n\times n}$ is unitary, $\mathbf{D} = diag(b_1, b_2, \dots, b_n)$ and $b_{max} = b_1 \geq b_2 \geq \cdots \geq b_n = b_{min} > 0$ are the eigenvalues of $\mathbf{B}$. The 2-norm of $\mathbf{B}$ is

$$\|\mathbf{B}\|_2 = |b_{max}|. \tag{20}$$

Substituting (19) into (15) yields

$$\mathbf{y} = \mathbf{Q}\mathbf{D}\mathbf{Q}^H\mathbf{s}.$$

Note that $\mathbf{y}, \mathbf{s} \in \mathbb{C}^{n\times 1}$ are vectors, it is known that

$$\begin{aligned}\mathbf{y}^H\mathbf{s} &= \mathbf{s}^H\mathbf{Q}\mathbf{D}^H\mathbf{Q}^H\mathbf{s}\\ &= (\mathbf{Q}^H\mathbf{s})^H\mathbf{D}^H(\mathbf{Q}^H\mathbf{s})\\ &= \mathbf{t}^H\mathbf{D}^H\mathbf{t}\\ &= \sum_{i=1}^{n} b_i|a_i|^2,\end{aligned} \tag{21}$$

where $\mathbf{t} = \mathbf{Q}^H\mathbf{s} \triangleq [a_1, a_2, \dots, a_n]^T$.

Similarly, we have

$$\begin{aligned}\mathbf{y}^H\mathbf{y} &= \mathbf{t}^H\mathbf{D}^H\mathbf{D}\mathbf{t}\\ &= \sum_{i=1}^{n} b_i^2|a_i|^2.\end{aligned} \tag{22}$$

Then, it follows from (21) and (22) that

$$\begin{aligned}|\lambda_2| &= \left|\frac{\sum_{i=1}^n b_i|a_i|^2}{\sum_{i=1}^n b_i^2|a_i|^2}\right|\\ &= \left|\frac{\sum_{i=1}^n w_i}{\sum_{i=1}^n w_i b_i}\right|,\end{aligned} \tag{23}$$

where $w_i = b_i|a_i|^2$.

From (23), $1/|\lambda_2|$ can be viewed as the weighted average of the eigenvalues, and the magnitude of weights is dependent on the eigenvalues. As discussed in Sagun et al. (2016), for neural networks, most of the eigenvalues of Hessian matrix of the loss function are close to zero. In this situation, one has

$$\frac{1}{|\lambda_2|} \approx \max_i |b_i|, \tag{24}$$

especially when the largest eigenvalue is much bigger than others. Therefore, according to (20), it is reasonable to take $\frac{1}{|\lambda_2|}$ as an approximate of $\|\mathbf{B}\|_2$.

While, let $\mathbf{s} = [s_1, s_2, \dots, s_n]^T$ and follow the above analysis, one can deduce that

$$|\lambda_1| = \left|\frac{\sum_{i=1}^n b_i|a_i|^2}{\sum_{i=1}^n |s_i|^2}\right|. \tag{25}$$

Note that $\mathbf{t} = \mathbf{Q}^H\mathbf{s}$ and $\mathbf{Q}$ is unitary, we have

$$\sum_{i=1}^{n} |s_i|^2 = \sum_{i=1}^{n} |a_i|^2, \tag{26}$$

and thus

$$|\lambda_1| = \left|\frac{\sum_{i=1}^n b_i|a_i|^2}{\sum_{i=1}^n |a_i|^2}\right|. \tag{27}$$

It can also be viewed as a weighted average of the eigenvalues. However, the weights are not dependent on the eigenvalues. So, it can only know from (27) that $b_{min} \leq \lambda_1 \leq b_{max}$ and it is obvious that $1/|\lambda_2|$ gives a better estimation to $\|\mathbf{B}\|_2$ than $|\lambda_1|$. Therefore, at the tth iteration, the stepsize is suggested to be

$$\lambda_t = \left|\frac{\mathbf{y}_t^H\mathbf{s}_t}{\mathbf{y}_t^H\mathbf{y}_t}\right|. \tag{28}$$

The flowchart of AS-CNAG is presented as Algorithm 1.

Algorithm 1 AS-CNAG

Initialization:

Given $\mathbf{w}_0$, α_0, $0 < \alpha_{min} < \alpha_{max}$, $\mathbf{v}_0 = 0, \beta < 0$;

$t \leftarrow 1$

repeat

$\hat{\mathbf{w}}_{t+1} = \mathbf{w}_t + \beta\mathbf{v}_{t-1}$

$\mathbf{g}_{t+1} = \nabla_{\mathbf{w}^*} J(\hat{\mathbf{w}}_{t+1})$

$\mathbf{y}_t = \mathbf{g}_{t+1} - \mathbf{g}_t$, $\mathbf{s}_t = \hat{\mathbf{w}}_{t+1} - \hat{\mathbf{w}}_t$

$\lambda_t = \left|\frac{\mathbf{y}_t^H\mathbf{s}_t}{\mathbf{y}_t^H\mathbf{y}_t}\right|$

$\alpha_t = \min\left\{\alpha_{max}, \max\left\{\alpha_{min}, \lambda_t\right\}\right\}$

$\mathbf{v}_t = \beta\mathbf{v}_{t-1} - \alpha_t\mathbf{g}_t$

$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_t$

$t \leftarrow t + 1$

until convergence

3.2. AS-CNAG-multi

According to multi-step quasi-Newton method (Ford & Moghrabi, 1994), by using the polynomial interpolation, the curvature information at previous iterations can be utilized to obtain a more accurate approximation of $\mathbf{B}$. Inspired by this, an improved version of Algorithm 1 can be established.

Let us consider the two-step (i.e., $m = 2$) update conditions

$$\begin{aligned}\mathbf{r}_t &= \mathbf{s}_t - \theta_t\mathbf{s}_{t-1},\\ \boldsymbol{\omega}_t &= \mathbf{y}_t - \theta_t\mathbf{y}_{t-1},\end{aligned} \tag{29}$$

where θ_t is a scaling parameter, which plays a crucial role in determining the extent to which the curvature information from the previous iteration (i.e., $\mathbf{s}_{t-1}$ and $\mathbf{y}_{t-1}$) influences the computation of $\mathbf{r}_t$ and $\boldsymbol{\omega}_t$. Note that (11) is derived from Lagrangian interpolation polynomial, and it does not limit the range of $\{\tau_j\}_{j=0}^m$. Therefore, the choice of θ_t is flexible.

With the help of the generalized quasi-Newton Eq. (10), CBBM can be extended to a general form which is called multi-step CBBM (MCBBM). In this situation, one has

$$\begin{aligned}\mu_{1,t} &= \arg\min_{\lambda} \left\|\frac{1}{\lambda}\mathbf{r}_t - \boldsymbol{\omega}_t\right\|^2\\ &= \frac{\mathbf{r}_t^H\boldsymbol{\omega}_t}{\mathbf{r}_t^H\mathbf{r}_t}\\ &= \frac{(\mathbf{s}_t - \theta_t\mathbf{s}_{t-1})^H(\mathbf{y}_t - \theta_t\mathbf{y}_{t-1})}{(\mathbf{s}_t - \theta_t\mathbf{s}_{t-1})^H(\mathbf{s}_t - \theta_t\mathbf{s}_{t-1})},\end{aligned} \tag{30}$$

which is an extension of (17). Similarly, (18) can be modified as

$$\mu_{2,t} = \frac{\boldsymbol{\omega}_t^H\mathbf{r}_t}{\boldsymbol{\omega}_t^H\boldsymbol{\omega}_t}. \tag{31}$$

Similar to Algorithm 1, let

$$\mu_t = \left|\frac{\boldsymbol{\omega}_t^H\mathbf{r}_t}{\boldsymbol{\omega}_t^H\boldsymbol{\omega}_t}\right| \tag{32}$$

be the stepsize of AS-CNAG-multi.

In the unit-spaced method, $\theta = \frac{1}{3}$. However, the performance of the unit-spaced method is not significantly improved compared to the traditional quasi-Newton method. Actually, when the loss function changes sharply within several iterations, using too much curvature information from the former iterations would lead to a large deviation

between complex Hessian matrix and its estimation. For example, as discussed in Ford and Moghrabi (1993), if the proportion of $\mathbf{s}_{t-1}$ in $\mathbf{r}_t$ is much larger than the one of $\mathbf{s}_t$, it yields that $\mathbf{r}_t \approx -\frac{1}{3}\mathbf{s}_{t-1}$. In this case, it is also possible that $\boldsymbol{\omega}_t \approx -\frac{1}{3}\mathbf{y}_{t-1}$. Then the generalized quasi-Newton Eq. (10) is degraded as

$$\mathbf{B}_{t+1}\mathbf{s}_{t-1} = \mathbf{y}_{t-1},$$

where the most recent curvature information (i.e., $\mathbf{s}_t$ and $\mathbf{y}_t$) is almost ignored. Thus, $\mathbf{B}_{t+1}$ would deviate largely from complex Hessian matrix at the current iteration compared with the original secant Eq. (15).

To overcome this disadvantage, θ should better be adaptive during the iteration process. A simple rule is that, to yield a satisfactory approximation of complex Hessian matrix, the curvature information of the current iteration (i.e., $\mathbf{s}_t$, $\mathbf{y}_t$) is more important than that of the former $\mathbf{s}_{t-1}$, $\mathbf{y}_{t-1}$. Thus, when the curvature information at the former iterations varies largely from the current one, the value of the scaling parameter θ should be decreasing. On the contrary, θ can be increased. Based on this, an improved version of Algorithm 1 is established and described as Algorithm 2.

Algorithm 2 AS-CNAG-multi

Initialization:

Given $\mathbf{w}_0$, α_0, ,$0 < \alpha_{min} < \alpha_{max}$, $\mathbf{v}_0 = 0, \mu_0 = 0, \beta < 0$;

$t \leftarrow 1$

1: **repeat**

2: $\hat{\mathbf{w}}_{t+1} = \mathbf{w}_t + \beta\mathbf{v}_{t-1}$

3: $\mathbf{g}_{t+1} = \nabla_{\mathbf{w}^*} J(\hat{\mathbf{w}}_{t+1})$

4: $\mathbf{y}_t = \mathbf{g}_{t+1} - \mathbf{g}_t$, $\mathbf{s}_t = \hat{\mathbf{w}}_{t+1} - \hat{\mathbf{w}}_t$

5: $\lambda_t = \left|\frac{\mathbf{y}_t^H \mathbf{s}_t}{\mathbf{y}_t^H \mathbf{y}_t}\right|$

6: $\gamma_t = \frac{\mu_{t-1}}{|\lambda_t - \mu_{t-1}|}$

7: $\theta_t = tanh(\gamma_t)$

8: $\boldsymbol{\omega}_t = \mathbf{y}_t - \theta_t \mathbf{y}_{t-1}, \mathbf{r}_t = \mathbf{s}_t - \theta_t \mathbf{s}_{t-1}$

9: $\mu_t = \left|\frac{\boldsymbol{\omega}_t^H \mathbf{r}_t}{\boldsymbol{\omega}_t^H \boldsymbol{\omega}_t}\right|$

10: $\alpha_t = \min\left\{\alpha_{max}, \max\left\{\alpha_{min}, \mu_t\right\}\right\}$

11: $\mathbf{v}_t = \beta\mathbf{v}_{t-1} - \alpha_t \mathbf{g}_t$

12: $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_t$

13: $t \leftarrow t + 1$

14: **until** convergence

According to MCBBM, μ_{t-1} is an approximation of curvature information at the former iteration to some extent. While, λ_t is an approximation of curvature information at the current iteration. So in Step 6, $\gamma_t > 0$ is employed to measure the inverse of the relative change between μ_{t-1} and λ_t. If $\mathbf{B}_{t+1}$ deviates heavily from $\mathbf{B}_t$, it is more likely that λ_t is far away from μ_{t-1}. Then, γ_t would be small and less curvature information at the former step is utilized. On the contrary, if μ_{t-1} and λ_t are close, γ_t would become larger and more curvature information at the former step should be utilized. On the other hand, to avoid θ_t to be too large such that the current curvature information is almost completely ignored, $tanh(\cdot)$ is adopted in Step 7 to ensure $\theta_t \in (0, 1)$.

4. Experiments

In this section, some experiments on pattern classification, nonlinear channel equalization, wind forecasting and SAR target classification are conducted to illustrate the effectiveness of AS-CNAG and AS-CNAG-multi.

4.1. Effect and ablative analysis of the momentum

We firstly take the Optical dataset in Table 1 as an example to analyze the effect of the momentum β. A fully CVNN with 20 hidden neurons is constructed for this classification task.

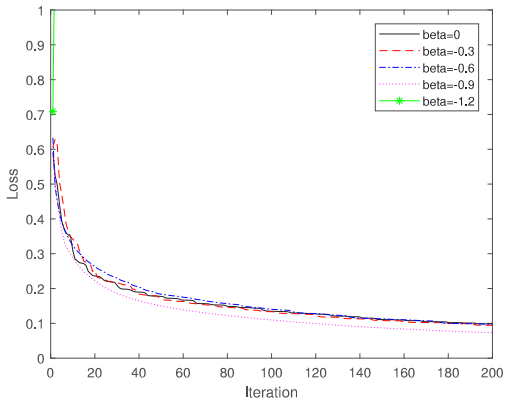


Fig. 1. The effect of the momentum on AS-CNAG.

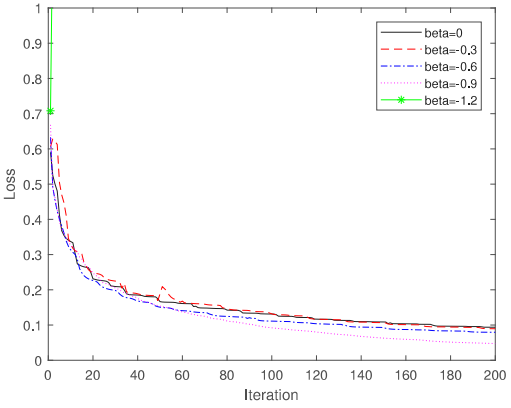


Fig. 2. The effect of the momentum term on AS-CNAG-multi.

Table 1
Summary of the datasets.

Datasets	Data size	Features	Classes
Spambase	4601	57	2
BUPA	345	6	2
Banknote	1372	4	2
Banana	5300	2	2
Glass	214	10	6
Optical	5620	64	10
Page blocks	5473	10	5
Waveform40	5000	40	3
Websitephishing	1353	9	3
Abalone	4177	8	3
Segmentation	2310	18	7
Robotnavigation	5456	25	4

For CNAG, the momentum β determines how much the complex gradient information is retained from previous iterations. When $\beta = 0$, CNAG is the traditional CGD method, and so are AS-CNAG and AS-CNAG-multi, which become adaptive stepsize based CGD.

It can be seen from Figs. 1 and 2 that, when $\beta = -0.9$, AS-CNAG and AS-CNAG-multi perform best. However, when $\beta = -1.2$, the training process is even divergent. Actually, β is empirically set as a value around -0.9 (Ruder, 2016). In subsequent experiments, if not specified, β defaults to -0.9. In addition, α_{min} and α_{max} are respectively set as 0.000001 and 10 by default.

4.2. Pattern classification

The basic information of the datasets used for pattern classification is described in Table 1 (Dua & Graff, 2017). The Spambase dataset is

Table 2
The test accuracy achieved by different algorithms.

Datasets	CGD	CBBM	CNAG	CALRT	AdGD-accel	AS-CNAG	AS-CNAG-multi
Spambase	0.609	0.629	0.729	0.865	0.845	0.898	**0.918**
BUPA	0.558	0.587	0.721	0.664	0.712	0.721	**0.760**
Banknote	0.896	0.917	0.968	0.976	0.956	**1.000**	**1.000**
Banana	0.549	0.588	0.581	0.710	0.724	0.829	**0.880**
Glass	0.797	0.813	0.813	0.813	0.813	0.844	**0.875**
Optical	0.909	0.908	0.941	0.917	0.927	0.934	**0.953**
Page blocks	0.910	0.910	0.910	0.928	0.928	0.911	**0.937**
Waveform40	0.851	0.857	0.865	0.867	0.868	0.867	**0.870**
Websitephishing	0.813	0.820	0.845	0.845	0.860	0.884	**0.887**
Abalone	0.542	0.535	0.548	0.542	0.546	0.554	**0.558**
Segmentation	0.772	0.801	0.896	0.889	0.895	0.900	**0.915**
Robotnavigation	0.634	0.690	0.725	0.768	0.795	0.854	**0.867**

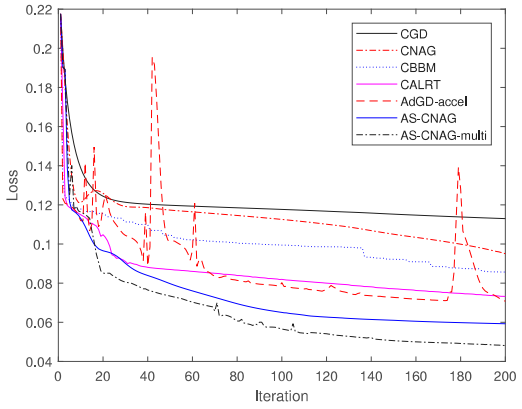


Fig. 3. The training losses of different methods on the Spambase dataset.

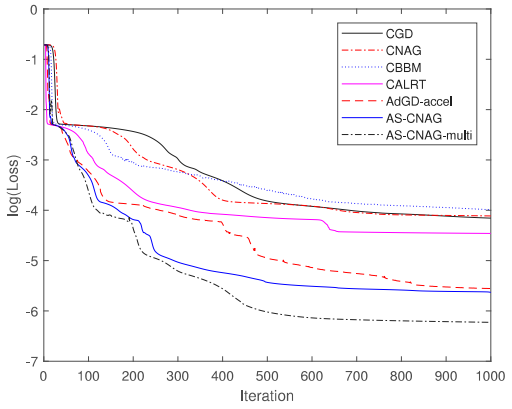


Fig. 4. The training losses of different methods for the channel equalization problem.

taken as an example to intuitively compare our methods with some previous algorithms including CGD, CNAG (with constant stepsize) (Zhang & Xia, 2015), CBBM (with line search) (Zhang & Mandic, 2015), CALRT (Zhang & Huang, 2020) and AdGD-accel, which is a complex-valued form of the algorithm in Malitsky and Mishchenko (2020). Note that AdGD-accel is a real-valued method, and is extended here as the complex-valued form for comparison. The number of training iterations is set as 200, and the loss curves of these methods for this task are presented in Fig. 3.

It can be clearly observed from Fig. 3 that AS-CNAG and AS-CNAG-multi converge faster than other methods on the Spambase dataset. In fact, compared with AdGD-accel, when employing AS-CNAG to estimate the local smoothness constant, the training of the fully CVNN is faster and more stable. Moreover, by introducing multi-step quasi-Newton technique in AS-CNAG, the performance can be further improved. It implies that the proposed method for the calculation of the scaling parameter θ_t in Algorithm 2 can actually work.

To further support the above conclusion, the test accuracy achieved by different algorithms on these classification tasks is summarized in Table 2. The results in Table 2 show that, for most of the datasets, AS-CNAG-multi guarantees the highest test accuracy among these algorithms, which is also superior to AS-CNAG. Compared to CNAG with fixed stepsize, our methods with adaptive stepsize perform much better.

4.3. Nonlinear channel equalization

In this subsection, a nonlinear channel equalization problem (Cha & Kassam, 1995) is considered. The output of the channel model is defined by

$$o_n = (0.34 - i0.27)s_n + (0.87 + i0.43)s_{n-1} + (0.34 - i0.21)s_{n-2}, \tag{33}$$

$$y_n = o_n + 0.1o_n^2 + 0.05o_n^3 + v_n, \tag{34}$$

where v_n is a white Gaussian noise and the signal-to-noise ratio is set to be 20 dB. s_n, s_{n-1} and s_{n-2} are respectively the input and delayed inputs of the model, where $s_n = [-0.7 - 0.7i, -0.7 + 0.7i, 0.7 - 0.7i, 0.7 + 0.7i]$ is a 4-QAM modulated signal. Let y_n and s_{n-2} be the output and the corresponding desired output of the fully CVNN, respectively.

In this experiment, the stepsizes of CGD and CNAG are both set as 0.1. The line search history length of CBBM is set to be 3. The scaling and rotation factors and the beam size of CALRT are $\{0.5, 1, 2\}$, $\{-\frac{\pi}{3}, 0, \frac{\pi}{3}\}$ and 5, respectively. AdGD-accel, AS-CNAG and AS-CNAG-multi use the default settings in Section 4.1. A fully CVNN with three input neurons, 30 hidden neurons and one output neuron is designed for this task and 5000 randomly generated samples are used to train it.

The training losses of different methods for this task are presented in Fig. 4. It is obvious that AS-CNAG-multi achieves the fastest convergence and the smallest loss. Indeed, AS-CNAG and AS-CNAG-multi significantly outperform the others in this example.

4.4. Complex-valued wind forecasting

In order to further demonstrate the efficiency of the two algorithms, a practical wind forecasting problem is considered (Mandic & Goh, 2009). The data used for wind forecasting are represented by the wind speed and direction averaged over one minute.[1]

Wind speed and direction are separately used as the magnitude and phase of the complex vector

$$w(t) = v(t)e^{id(t)},$$

[1] The dataset is obtained from iastate.edu. The Andover station is chosen, and the dataset contains the wind speed and direction observed in January 2022.

Table 3
Performance of different methods on the wind forecasting problem.

Algorithms	MAE	nRMSE	R^2
CGD	4.80%	26.78%	92.46%
CBBM	4.80%	26.80%	92.45%
CNAG	4.83%	26.97%	92.35%
CALRT	4.46%	25.01%	93.43%
AdGD-accel	3.97%	22.70%	94.58%
AS-CNAG	3.85%	21.86%	94.98%
AS-CNAG-multi	**3.84%**	**21.78%**	**95.01%**

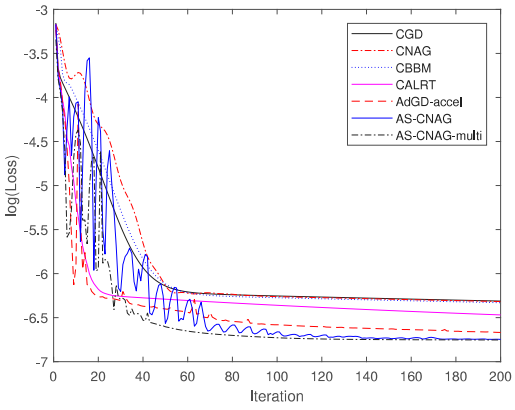


Fig. 5. The training losses of different methods for the wind forecasting problem.

where $w(t)$ is the combined complex vector, $v(t)$ and $d(t)$ are respectively the wind speed and direction.

For the wind dataset, the sth to $(s+2)$th elements (i.e., $w(s), w(s+1)$ and $w(s+2)$) are combined as the sth sample, and the $(s+3)$th one is the corresponding desired output, where $s = 1, 2, \ldots, 4997$. Here, 3000 samples of them are used as the training set and the rest ones are used as the test set.

In this experiment, the stepsizes of CGD and NAG are respectively set as 1 and 0.1. The line search history length of CBBM is set to be 3. The setting of other algorithms follows the previous one in Section 4.3. A fully CVNN with three input neurons, 30 hidden neurons and one output neuron is designed for this task.

In order to compare the forecasting performance of different algorithms, three quantitative performance criteria are taken into account including the mean absolute error (MAE), the normalized root mean squared error (nRMSE) and the coefficient of determination R^2 (Saad et al., 2017), which are respectively defined by

$$\mathrm{MAE} = \frac{\sum_{j=1}^{J} \left|\mathbf{o}_j - \mathbf{y}_j\right|}{J},$$

$$\mathrm{nRMSE} = \frac{\sqrt{\frac{1}{J}\sum_{j=1}^{J} \left|\mathbf{o}_j - \mathbf{y}_j\right|^2}}{\sqrt{\frac{1}{J}\sum_{j=1}^{J} \left|\mathbf{y}_j\right|^2}},$$

$$R^2 = 1 - \frac{\sum_{j=1}^{J} \left|\mathbf{o}_j - \mathbf{y}_j\right|^2}{\sum_{j=1}^{J} \left|\mathbf{o}_j - \bar{\mathbf{y}}\right|^2},$$

where J is the number of samples and $\bar{\mathbf{y}}$ is the mean of the desired output.

Fig. 5 shows that, during the training process, the AS-CNAG-multi is faster than the others. During the first 100 iterations, AS-CNAG fluctuates wildly, while AS-CNAG-multi performs better. It implies that, with a more accurate approximate to the norm of complex Hessian matrix, not only faster convergence can be achieved, but also more stable training can be guaranteed. Besides, the prediction performance on the test set of the seven methods are summarized in Table 3, which clearly shows that AS-CNAG and AS-CNAG-multi perform better than the compared methods.

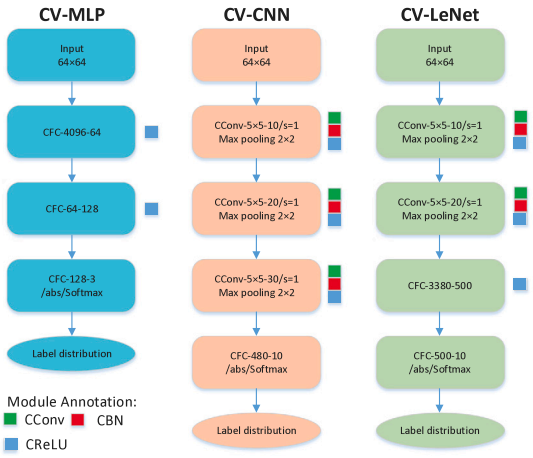


Fig. 6. Architectures of CV-MLP, CV-CNN and CV-LeNet.

4.5. SAR target classification

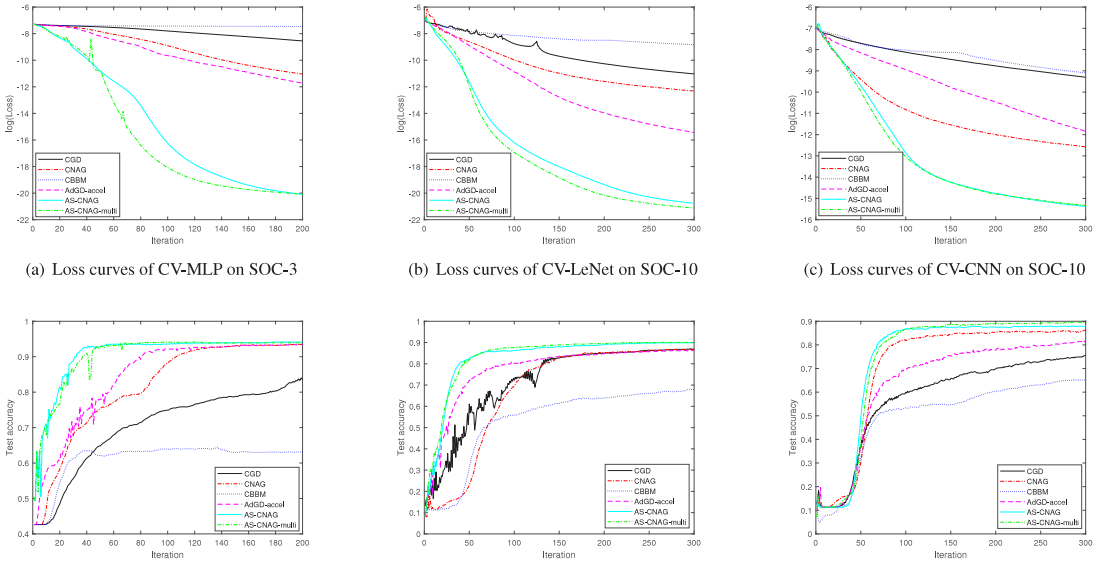
In this subsection, our algorithms are further employed to train complex-valued multi-layer perceptron (CV-MLP) and complex-valued convolutional neural networks (CV-CNN) for the purpose of SAR target classification of the moving and stationary target acquisition and recognition (MSTAR) dataset (Keydel et al., 1996). It contains complex-valued 2D SAR images and the region of interest is carefully chosen from the SAR samples with the dimension of 64 × 64. To evaluate the performance of different algorithms, both the standard operating condition (SOC) dataset and the extended operating condition (EOC) dataset from MSTAR are utilized (Zeng et al., 2022). Specifically, the SOC dataset comprises of the SOC-10 and SOC-3 subsets, while the EOC dataset includes the depression variation (EOC-NV-10) dataset. Detailed information regarding these datasets can be found in Table 4. It should be noted that, in the EOC-NV-10 dataset, the training samples primarily belong to the training set of SOC-10. While, its test set is taken from the test set of SOC-10 subject to different levels of additive white Gaussian noise with the signal-to-noise (SNR) varying from −10 dB to 10 dB.

The experimental models employed in this study include CV-MLP, CV-CNN (Zhang et al., 2017) and CV-LeNet (Trabelsi et al., 2017). The architectural representations of these models can be observed from Fig. 6. The CV-MLP depicted in Fig. 6 consists of two hidden layers with 64 and 128 neurons, respectively. In both CV-CNN and CV-LeNet, the convolutional layers employ a kernel size of 5 with a stride of 1. Following each convolutional layer, a 2 × 2 pooling layer is connected. Besides, in the CV-CNN, there are three convolutional layers and one fully connected layer. The convolutional layers consist of 10, 20 and 30 filters, respectively. The output layer is a fully connected layer with 10 output neurons. On the other hand, CV-LeNet comprises of two convolutional layers and two fully connected layers. The numbers of filters in the convolutional layers are 10 and 20, respectively. The fully connected layers have 500 and 10 neurons, respectively. In the following experiments, the stepsize of CGD and CNAG is fixed as 0.01. As for the AdGD-accel, CBBM, AS-CNAG and AS-CNAG-multi algorithms, their maximum stepsize (the modulus of the complex-valued stepsize for CBBM) is set to 0.1, while the minimum stepsize is set to 1e-8. It is pointed out that, due to the excessive computational complexity of CALRT, it is not suitable for training deep CVNNs. Therefore, it is excluded for comparison from the subsequent experiments.

The classification of the SOC-3 dataset is performed using the CV-MLP in Fig. 6, and the SOC-10 dataset is classified using CV-CNN

Table 4
Configurations of the EOC and SOC subsets.

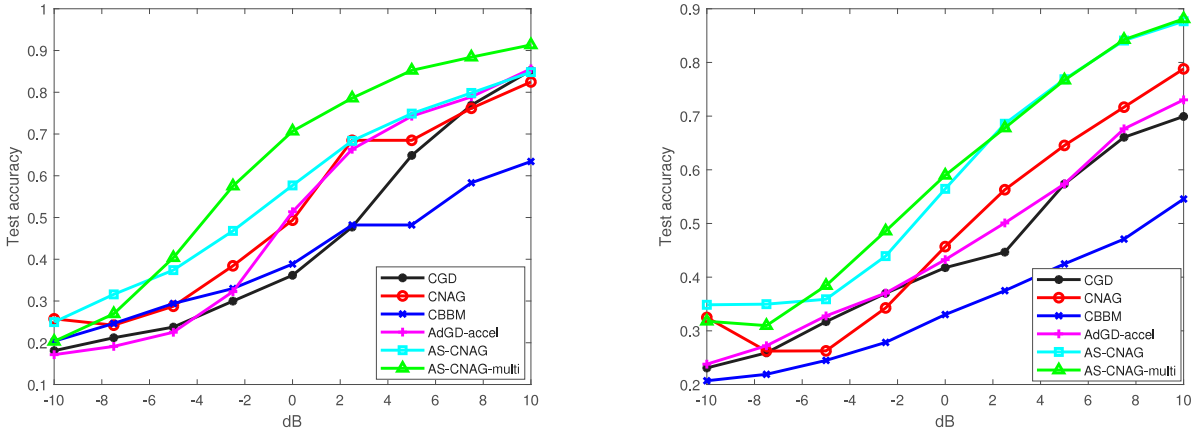
Datasets	Target types	Number of training samples	Number of test samples	Classes
SOC-3	2-BMP2(9563),2-BMP2(9566),2-BMP2(c21), 5-BTR70,8-T72(132),8-T72(812),8-T72(s7)	1622	1365	3
SOC-10	1-2S1,2-BMP2(9563),3-BRDM2,4-BTR60,5-BTR70, 6-D7,7-T62,8-T72(132),9-ZIL131,10-ZSU234	2747	2425	10
EOC-NV-10	1-2S1,2-BMP2(9563),3-BRDM2,4-BTR60,5-BTR70, 6-D7,7-T62,8-T72(132),9-ZIL131,10-ZSU234	2747	2425	10



(a) Loss curves of CV-MLP on SOC-3 (b) Loss curves of CV-LeNet on SOC-10 (c) Loss curves of CV-CNN on SOC-10

(d) Test accuracy curves of CV-MLP on SOC-3 (e) Test accuracy curves of CV-LeNet on SOC-10 (f) Test accuracy curves of CV-CNN on SOC-10

Fig. 7. Loss and accuracy curves of different algorithms on the SOC-3 and SOC-10 datasets.



(a) Test accuracy of CV-LeNet on EOC-NV-10 (b) Test accuracy of CV-CNN on EOC-NV-10

Fig. 8. Test accuracy of the compared algorithms on the EOC-NV-10 dataset with different SNRs.

and CV-LeNet. The experimental results are illustrated in Fig. 7. Upon analyzing Fig. 7, it becomes apparent that the models trained by AS-CNAG and AS-CNAG-multi algorithms exhibit a more rapid reduction of the loss value and guarantee more higher test accuracy than by CGD, CNAG, CBBM and AdGD-accel.

The experimental results on the EOC-NV-10 dataset are shown in Fig. 8. Clearly, with the SNR of 10 dB, all algorithms achieve the best performance. As the SNR gradually decreases, the recognition accuracy of CV-CNN and CV-LeNet trained by these algorithms progressively declines. Moreover, for most of the cases, the models trained by AS-CNAG-multi obtain the highest test accuracies. It verifies the effectiveness and robustness of AS-CNAG-multi over some previous ones for the training of CV-CNN and CV-LeNet.

5. Conclusion

In this paper, two adaptive stepsize based CNAG methods have been proposed for training fully CVNNs. In our methods, the stepsize of CNAG has been adaptively designed as the inverse of local smoothness constant of the loss function, which is an estimation of

the norm of approximate Hessian matrix $\|\mathbf{B}\|$. Theoretical analysis has been presented to confirm the feasibility of the estimation method. To estimate $\|\mathbf{B}\|$ more accurately, an efficient method for the calculation of the scaling parameter in multi-step quasi-Newton update conditions has been introduced to further improve the performance of AS-CNAG. Experimental results on pattern classification, nonlinear channel equalization, wind forecasting and SAR target classification problems have been provided to demonstrate the effectiveness of the proposed methods. It has verified that much better performance can be achieved by our methods than by some existing ones.

CRediT authorship contribution statement

Weijing Zhao: Methodology, Software, Validation, Writing – original draft. **He Huang:** Conceptualization, Methodology, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments that have greatly improved the quality of this paper. This work was jointly supported by Qinglan Project of Jiangsu Province, China, the Natural Science Foundation of Jiangsu Province of China under Grant no. BK20181431 and the Postgraduate Research and Practice Innovation Program of Jiangsu Province, China under grant no. KYCX23_3247.

References

Aizenberg, I. (2011). *Complex-valued neural networks with multi-valued neurons*. Springer.
Barzilai, J., & Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, *8*(1), 141–148.
Beck, A. (2017). *First-order methods in optimization*. SIAM.
Ceylan, R., Ceylan, M., Özbay, Y., & Kara, S. (2011). Fuzzy clustering complex-valued neural network to diagnose cirrhosis disease. *Expert Systems with Applications*, *38*(8), 9744–9751.
Cha, I., & Kassam, S. A. (1995). Channel equalization using adaptive complex radial basis function networks. *IEEE Journal on Selected Areas in Communications*, *13*(1), 122–131.
Chen, H., Natsuaki, R., & Hirose, A. (2022). Polarization-aware prediction of mobile radio wave propagation based on complex-valued and quaternion neural networks. *IEEE Access*, *10*, 66589–66600.
Dozat, T. (2016). Incorporating Nesterov momentum into Adam. In *Proceedings of the international conference on learning representations* (pp. 2013–2016).
Dua, D., & Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml.
Ford, J. (2001). Implicit updates in multistep quasi-Newton methods. *Computers & Mathematics with Applications*, *42*(8–9), 1083–1091.
Ford, J., & Moghrabi, I. (1993). Alternative parameter choices for multi-step quasi-Newton methods. *Optimization Methods & Software*, *2*(3–4), 357–370.
Ford, J., & Moghrabi, I. (1994). Multi-step quasi-Newton methods for optimization. *Journal of Computational and Applied Mathematics*, *50*(1–3), 305–323.
Horn, R. A., & Johnson, C. R. (2012). *Matrix analysis*. Cambridge University Press.
Keydel, E. R., Lee, S. W., & Moore, J. T. (1996). MSTAR extended operating conditions: A tutorial. In *Algorithms for synthetic aperture radar imagery III, Vol. 2757* (pp. 228–242). SPIE.
Kreutz-Delgado, K. (2009). The complex gradient operator and the CR-calculus. ArXiv, abs/0906.4835.
Leung, H., & Haykin, S. (1991). The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, *39*(9), 2101–2104.
Li, H., & Adalı, T. (2008). Complex-valued adaptive signal processing using nonlinear functions. *EURASIP Journal on Advances in Signal Processing*, *2008*, 1–9.
Malitsky, Y., & Mishchenko, K. (2020). Adaptive gradient descent without descent. In *Proceedings of the 37th international conference on machine learning* (pp. 6702–6712).
Mandic, D. P., & Goh, V. S. L. (2009). *Complex valued nonlinear adaptive filters: Noncircularity, widely linear and neural models*. John Wiley & Sons.
Marsden, J. E., & Hoffman, M. J. (1993). *Elementary classical analysis*. Macmillan.
Moghrabi, I. A. (2017). Implicit extra-update multi-step quasi-Newton methods. *International Journal of Operational Research*, *28*(2), 216–228.
Moughrabi, I. A. (2009). New implicit multistep quasi-Newton methods. *Numerical Analysis and Applications*, *2*(2), 154–164.
Nesterov, Y. E. (1983). A method of solving a convex programming problem with convergence rate $o(k^2)$. In *Doklady akademii nauk, Vol. 269* (pp. 543–547). Russian Academy of Sciences.
Nishioka, A., & Kanno, Y. (2021). Accelerated projected gradient method with adaptive step size for compliance minimization problem. *JSIAM Letters*, *13*, 33–36.
Nitta, T. (2004). Orthogonality of decision boundaries in complex-valued neural networks. *Neural Computation*, *16*(1), 73–97.
Nitta, T. (2013). Local minima in hierarchical structures of complex-valued neural networks. *Neural Networks*, *43*, 1–7.
Peker, M. (2021). Classification of hyperspectral imagery using a fully complex-valued wavelet neural network with deep convolutional features. *Expert Systems with Applications*, *173*, Article 114708.
Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, *4*(5), 1–17.
Rawat, S., Rana, K., & Kumar, V. (2021). A novel complex-valued convolutional neural network for medical image denoising. *Biomedical Signal Processing and Control*, *69*, Article 102859.
Rojc, M., & Mlakar, I. (2020). A new fuzzy unit selection cost function optimized by relaxed gradient descent algorithm. *Expert Systems with Applications*, *159*, Article 113552.
Ruder, S. (2016). An overview of gradient descent optimization algorithms. ArXiv, abs/1609.04747.
Saad, L., Rahmoune, F., Tourtchine, V., & Baddari, K. (2017). Fully complex valued wavelet network for forecasting the global solar irradiation. *Neural Processing Letters*, *45*(2), 475–505.
Sagun, L., Bottou, L., & LeCun, Y. (2016). Eigenvalues of the Hessian in deep learning: Singularity and beyond. ArXiv, abs/1611.07476.
Scarnati, T., & Lewis, B. (2021). Complex-valued neural networks for synthetic aperture radar image classification. In *Proceedings of the 2021 IEEE radar conference* (pp. 1–6).
Singhal, U., Xing, Y., & Yu, S. X. (2022). Co-domain symmetry for complex-valued deep learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 681–690).
Sorber, L., Barel, M. V., & Lathauwer, L. D. (2012). Unconstrained optimization of real functions in complex variables. *SIAM Journal on Optimization*, *22*(3), 879–898.
Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., & Pal, C. J. (2017). Deep complex networks. ArXiv, abs/1705.09792.
Virmaux, A., & Scaman, K. (2018). Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Advances in neural information processing systems, Vol. 31*.
Wirtinger, W. (1927). Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen*, *97*(1), 357–375.
Xie, W., Ma, G., Zhao, F., Liu, H., & Zhang, L. (2020). PolSAR image classification via a novel semi-supervised recurrent complex-valued convolution neural network. *Neurocomputing*, *388*, 255–268.
Yang, T., Lin, Q., & Li, Z. (2016). Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. ArXiv, abs/1604.03257.
Zeng, Z., Sun, J., Han, Z., & Hong, W. (2022). SAR automatic target recognition method based on multi-stream complex-valued networks. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–18.
Zhang, Y., & Huang, H. (2020). Adaptive complex-valued stepsize based fast learning of complex-valued neural networks. *Neural Networks*, *124*, 233–242.
Zhang, B., Liu, Y., Cao, J., Wu, S., & Wang, J. (2019). Fully complex conjugate gradient-based neural networks using Wirtinger calculus framework: Deterministic convergence and its application. *Neural Networks*, *115*, 50–64.
Zhang, H., Liu, X., Xu, D., & Zhang, Y. (2014). Convergence analysis of fully complex backpropagation algorithm based on Wirtinger calculus. *Cognitive Neurodynamics*, *8*(3), 261–266.
Zhang, H., & Mandic, D. P. (2015). Is a complex-valued stepsize advantageous in complex-valued gradient learning algorithms? *IEEE Transactions on Neural Networks and Learning Systems*, *27*(12), 2730–2735.
Zhang, Z., Wang, H., Xu, F., & Jin, Y.-Q. (2017). Complex-valued convolutional neural network and its application in polarimetric SAR image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(12), 7177–7188.
Zhang, S., & Xia, Y. (2015). Two fast complex-valued algorithms for solving complex quadratic programming problems. *IEEE Transactions on Cybernetics*, *46*(12), 2837–2847.
Zhang, H., Zhang, C., & Wu, W. (2009). Convergence of batch split-complex backpropagation algorithm for complex-valued neural networks. *Discrete Dynamics in Nature and Society*, *2009*, 1–16.