Chaos, Solitons and Fractals 100 (2017) 57–61

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

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

Chaos time-series prediction based on an improved recursive Levenberg–Marquardt algorithm

Xiancheng Shi[a,b], Yucheng Feng[a,b,*], Jinsong Zeng[a,b], Kefu Chen[a,b]

[a] *School of Light Industry and Engineering, South China University of Technology, Guangdong, 510641, China*
[b] *State Key Laboratory of Pulp and Paper Engineering, Guangdong, 510641, China*

ARTICLE INFO

Article history:
Received 14 December 2016
Revised 10 April 2017
Accepted 18 April 2017

Keywords:
Recursive algorithm
Levenberg–Marquardt
On-line learning
Neural networks

ABSTRACT

An improved recursive Levenberg–Marquardt algorithm (RLM) is proposed to more efficiently train neural networks. The error criterion of the RLM algorithm was modified to reduce the impact of the forgetting factor on the convergence of the algorithm. The remedy to apply the matrix inversion lemma in the RLM algorithm was extended from one row to multiple rows to improve the success rate of the convergence; after that, the adjustment strategy was modified based on the extended remedy. Finally, the performance of this algorithm was tested on two chaotic systems. The results show improved convergence.

1. Introduction

The steepest descent method based on error back-propagation (SDBP) is generally regarded as the most common algorithm among all the supervised learning algorithms for training neural networks. However, major drawbacks of SDBP include its slow convergence speed and tendency to be entrapped in a local optimum. Because of the fast and stable convergence, the Levenberg–Marquardt (LM) algorithm has received widespread attention for producing a search direction between the Gauss-Newton and the steepest descent directions among the improved algorithms of SDBP [1]. However, the limitation on the computational time and storage space of the LM algorithm will still be problems when there a large number of training samples and node weights. Therefore, it is necessary to explore the potential of the LM algorithm to overcome these problems.

Many studies have been carried out on the off-line LM algorithm for training neural networks aimed at less computation and storage [2–4], improved convergence properties [5,6], and generalization performance [7,8]. An on-line LM algorithm is highly desirable when the process to be modeled is time varying or when it is impossible to obtain sufficient off-line data. To explore the potential of the LM algorithm, a recursive LM algorithm for the on-line training of neural networks has been proposed [9,10]. By neglecting interneuron weight correlations, RLM algorithms can be decomposed at the neuron level, enabling weights to be updated in an efficient parallel manner [11]. Among RLM algorithms, a temporal difference RLM has been developed to improve the action-dependent adaptive critic performance in terms of convergence and parameter shadowing [12]; a general recursive Bayesian Levenberg–Marquardt algorithm was also derived to obtain better generalization and stable numerical performance [13]. An on-line implementation of the LM algorithm based on a sliding window was developed for overcoming the difficulties in the implementation of the iterative version [14,15], in which a batch-sliding window with early stopping was applied. Compared to the recursive LM algorithm, the on-line LM algorithm based on a sliding window needs more storage space and has a higher computational cost, while the two main drawbacks of the RLM are the slow convergence speed due to the forgetting factor and instability caused by the remedy to apply the matrix inversion lemma.

The neural networks have performed well in the prediction of nonlinear and chaotic time series, they can present the most accurate one-step or multi-step ahead predictions [16–19]. In this paper, we propose an improved recursive Levenberg–Marquardt algorithm to improve the performance of the RLM algorithm in terms of convergence and apply the algorithm to online training of neural networks for prediction of chaotic time-series.

The paper is organized as follows. In Section 2, the off-line Levenberg–Marquardt algorithm is described. Section 3 presents the improved RLM algorithm, including a complete derivation of the improved RLM algorithm based on a corrected error criterion, the extended remedy to apply the matrix inversion lemma in the

* Corresponding author at: School of Light Industry and Engineering, South China University of Technology, Guangdong, 510641, China.

E-mail addresses: xianchengshi@126.com (X. Shi), fengyc@scut.edu.cn (Y. Feng), fezengjs@scut.edu.cn (J. Zeng), ppchenkf@scut.edu.cn (K. Chen).

http://dx.doi.org/10.1016/j.chaos.2017.04.032
0960-0779/© 2017 Elsevier Ltd. All rights reserved.

improved RLM algorithm, and the modified adjustment strategy based on the extended remedy. Section 4 presents the experiment results, and finally, a brief conclusion is provided in Section 5.

2. Levenberg–Marquardt algorithm

Neural networks can be used to approximate any complicated nonlinear mapping thanks to the properties of parallel computing and self-learning. Recurrent neural networks (RNNs) are widely used for system identification and control because of their good dynamic performance in terms of memory and feedback of the intermediate states of the RNNs. The performance index of the RNNs is defined as

$$V(t,\mathbf{W}) = \frac{1}{2}\mathbf{e}^T(t,\mathbf{W})\mathbf{e}(t,\mathbf{W}) = \frac{1}{2n_P}\sum_{r=1}^{n_P}\sum_{k=1}^{n_O}\left(T_{r,k}(t) - O_{r,k}(t,\mathbf{W})\right)^2 \tag{1}$$

where $\mathbf{e}(t, \mathbf{W})$ is a residual error vector with the form $[e_{1,1}(t,\mathbf{W}),\cdots,e_{1,n_O}(t,\mathbf{W}),\cdots,e_{n_P,1}(t,\mathbf{W}),\cdots,e_{n_P,n_O}(t,\mathbf{W})]^T$, $\mathbf{W}$ is the node weight vector with the form $[W_1, \cdots, W_d]^T$, n_P and n_O are the sample size and output layer node number, d is the total number of weights of neural networks, and $O_{r,k}(t, \mathbf{W})$ and $T_{r,k}(t)$ are the outputs and the corresponding targets.

The LM algorithm can be expressed as

$$\begin{aligned}\Delta\mathbf{W} &= -[\nabla^2\mathbf{V}(t,\mathbf{W})]^{-1}\nabla\mathbf{V}(t,\mathbf{W}) \\ &= -[\mathbf{J}(t,\mathbf{W})^T\mathbf{J}(t,\mathbf{W}) + \delta\mathbf{I}]^{-1}\mathbf{J}^T(t,\mathbf{W})\mathbf{e}_k(t,\mathbf{W})\end{aligned} \tag{2}$$

where $\Delta\mathbf{W}$ is the change in $\mathbf{W}$, $\nabla^2\mathbf{V}(t, \mathbf{W})$ and $\nabla\mathbf{V}(t, \mathbf{W})$ are the Hessian matrix and the first derivative matrix, $\mathbf{J}(t, \mathbf{W})$ is Jacobian matrix, and δ is the regularized parameter of LM algorithm. Further details are provided in [1].

3. The modified recursive Levenberg–Marquardt algorithm

The LM algorithm (6) is unable to incorporate new measurements without reprocessing the entire batch [15], and it cannot be applied to system identification, adaptive control, or time-series modeling of systems that are time-variant or real-time. An online learning algorithm is necessary to effectively adapt the changes of the dynamic systems. The recursive LM algorithm is an online algorithm that has been proposed for the online training of neural networks and other learning and self-adaption problems.

3.1. Corrected error criterion

The error criterion of the RLM algorithm was defined in [14] as follows

$$V(t,\mathbf{W}) = \frac{1}{2}\sum_{\tau}^{t}\lambda^{t-\tau}\mathbf{e}^2(\tau,\mathbf{W}) \tag{7}$$

where $0 < \lambda \le 1$ is the forgetting factor, which decides how fast the relevance of past data should decrease. It is the sum of the forgetting coefficient in Eq. (7) that will virtually slow down the convergence speed of the RLM algorithm and that must be removed to improve the convergence speed of the RLM algorithm. The sum of the forgetting coefficient is

$$\frac{1-\lambda^t}{1-\lambda} = \lambda^0 + \cdots + \lambda^{t-1}. \tag{8}$$

Thus, the error criterion should be corrected as

$$V_*(t,\mathbf{W}) = \frac{1-\lambda}{1-\lambda^t}V(t,\mathbf{W}) = \frac{1}{2}\frac{1-\lambda}{1-\lambda^t}\sum_{\tau=1}^{t}\lambda^{t-\tau}\mathbf{e}^2(\tau,\mathbf{W}). \tag{9}$$

Differentiating (9) with respect to $\mathbf{W}$ gives

$$\begin{aligned}\mathbf{V}'_*(t,\mathbf{W}) &= -\frac{1-\lambda}{1-\lambda^t}\sum_{\tau=1}^{t}\lambda^{t-\tau}\mathbf{J}^T(\tau,\mathbf{W})\mathbf{e}(\tau,\mathbf{W}) \\ &= \frac{1-\lambda^{t-1}}{1-\lambda^t}\lambda\mathbf{V}'_*(t-1,\mathbf{W}) - \frac{1-\lambda}{1-\lambda^t}\mathbf{J}^T(t,\mathbf{W})\mathbf{e}(t,\mathbf{W}),\end{aligned} \tag{10}$$

and differentiating once more

$$\begin{aligned}\mathbf{V}''_*(t,\mathbf{W}) = &\frac{1-\lambda^{t-1}}{1-\lambda^t}\lambda\mathbf{V}''_*(t-1,\mathbf{W}) + \frac{1-\lambda}{1-\lambda^t}\mathbf{J}^T(t,\mathbf{W})\mathbf{J}(t,\mathbf{W}) \\ &- \frac{1-\lambda}{1-\lambda^t}\frac{\partial^2 e(t,\mathbf{W})}{\partial\mathbf{W}\partial\mathbf{W}}\mathbf{e}(t,\mathbf{W}),\end{aligned} \tag{11}$$

multiplying (11) with $(1-\lambda^t)$ gives

$$\begin{aligned}(1-\lambda^t)\mathbf{V}''_*(t,\mathbf{W}) = &\lambda(1-\lambda^{t-1})\mathbf{V}''_*(t-1,\mathbf{W}) \\ &+ (1-\lambda)\mathbf{J}^T(t,\mathbf{W})\mathbf{J}(t,\mathbf{W}) \\ &- (1-\lambda)\frac{\partial^2\mathbf{e}(t,\mathbf{W})}{\partial\mathbf{W}\partial\mathbf{W}}\mathbf{e}(t,\mathbf{W}).\end{aligned} \tag{12}$$

If it is assumed that $\mathbf{W}$ is indeed the optimal estimate at time t, then $\partial^2\mathbf{e}(t, \mathbf{W})/(\partial\mathbf{W}\partial\mathbf{W}) \approx 0$. As $\mathbf{J}(t, \mathbf{W})^T\mathbf{J}(t, \mathbf{W})$ is a symmetrical semi-definite matrix, a regularized unit matrix $\delta\mathbf{I}$ was added to guarantee the approximate Hessian matrix positive definite [9]. Choosing $\mathbf{H}(t) = (1-\lambda^t)\mathbf{V}''_*(t,\mathbf{W}) + \delta\mathbf{I}$ to normalize $\mathbf{V}''_*(t, \mathbf{W})$ in Eq. (12), the improved RLM algorithm is

$$\mathbf{H}(t+1) = \lambda\mathbf{H}(t) + (1-\lambda)(\mathbf{J}(t,\mathbf{W})^T\mathbf{J}(t,\mathbf{W}) + \delta\mathbf{I}) \tag{13a}$$

$$\Delta\mathbf{W} = -(1-\lambda^t)\mathbf{H}^{-1}(t)\mathbf{J}^T(t,\mathbf{W})\mathbf{e}(t,\mathbf{W}). \tag{13b}$$

The RLM algorithm based on the error criterion (7) in [9,10] can be expressed as

$$\mathbf{H}(t+1) = \lambda\mathbf{H}(t) + (1-\lambda)(\mathbf{J}(t,\mathbf{W})^T\mathbf{J}(t,\mathbf{W}) + \delta\mathbf{I}) \tag{14a}$$

$$\Delta\mathbf{W} = -(1-\lambda)\mathbf{H}^{-1}(t)\mathbf{J}^T(t,\mathbf{W})\mathbf{e}(t,\mathbf{W}) \tag{14b}$$

where $\mathbf{H}(t) = (1-\lambda)\mathbf{V}''(t,\mathbf{W}) + \delta\mathbf{I}$. The difference between Eqs. (13a,b) and (14a,b) are the coefficients $(1-\lambda)$ and $(1-\lambda^t)$; the latter is close to 1, while the former is fixed when t increases continuously. When $\lambda > 0.5$, $\Delta\mathbf{W}$ in Eq. (13b) is at least twice as much as $\Delta\mathbf{W}$ in Eq. (14b), which will result in a better convergence property in the improved RLM algorithm; while $\lambda \le 0.5$, the RLM algorithm will have a similar convergence with the improved RLM algorithm. In [11], it is suggested that the forgetting factor λ range from 0.95 to 0.99, while in [13], a fast and accurate convergence is observed when starting with $\lambda = 0.90$. By reducing the impact of the forgetting factor in Eq. (13a,b), we optimize the range of λ, to be revealed in Section 4.

3.2. Matrix inversion

The computation of a direct matrix inversion of Eq. (13a) is $O(n_W^3)$, which is time consuming when there are so many parameters in the applied system. The matrix inversion lemma [20] can be used to reduce the computation complexity to $O(n_W^2)$, and the lemma expresses the following inversion

$$[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1}. \tag{15}$$

Unfortunately, the matrix inversion lemma cannot be applied to the improved RLM (13a) algorithm directly, and a possible remedy