Journal of Computational Information Systems 8: 7 (2012) 2915–2920
Available at http://www.Jofcis.com

A Neural-network Method Based on RLS Algorithm for Solving Special Linear Systems of Equations

Zhezhao ZENG [1,*], Yuxiang TU [2], Jinjin XIAO [1]

[1] *College of Electrical and Information Engineering, Changsha University of Science and Technology, Changsha 410014, China*

[2] *College of Computer and Communication Engineering, Hunan University, Changsha 410086, China*

Abstract

A neural-network method based on RLS algorithm (NN-RLS) is presented for solving special linear system of equations, such as Toeplitz, Vandermonder, Pascal, and Comrade systems, etc. The main idea of the approach is that special systems of linear equations are factored in the form of linear algebraic equations (LAE), and then neural-network model is constructed by LAE. The Recursive Least-Square (RLS) algorithm is used to train the weight vector of the neural network, which is the solution of system of LAE. The results reveal that the NN-RLS method is very simple and effective.

Keywords: Neural Network; RLS Algorithm; Special Linear System

1 Introduction

Consider the numerical solution of special system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{A} \in R^{n\times n}, and\ \mathbf{x}, \mathbf{b} \in R^{n} \tag{1}$$

where the coefficient matrix $\mathbf{A}$ is special matrix, such as Toeplitz, Vandermonder, Pascal, or Comrade matrices, etc. The computation of the system (1) is usually applied to the problems, such as binomial expansion, filter design [1, 2], probability, combinatorics, linear algebra [3], electrical engineering [4] and structure mechanics, model prediction [5] etc., and has important practical significations. In [6] M. Morháč proposed the error-free algorithms to special linear systems, e.g. Toeplitz, Hilbert, and Vandermonder systems. The main significance of the method is eliminating rounding-off and truncation errors. In recent years, the linear system of equations with coefficient matrices of Pascal type have been quietly investigated in different fields of applied linear algebra [7-10]. In [11] X.G. Lv presented an algorithm for solving linear systems of the Pascal matrices, which is based on the explicit factorization of the Pascal matrices. In [12] T. Sogabe proposed two algorithms for solving the comrade linear systems based on tridiagonal

*Corresponding author.
Email address: hncs66999@yahoo.com.cn (Zhezhao ZENG).

April 2012

solvers. In [13] A.A. Karawia presented two efficient algorithms for solving comrade linear systems. The two algorithms are based on the LU decomposition of the comrade matrix.

In this paper, we present a neural-network method based on RLS algorithm (NN-RLS) for solving problems (1). In the new method, we firstly transform system of (1) to the form of linear algebraic equations (LAE), and then neural-network model is constructed by LAE. The Recursive Least-Square (RLS) algorithm is used to train the weight vector of the neural network, which is the solution of system of (1). The numerical experiments show that the new NN-RLS method is very simple and effective.

2 The Description of Neural-network Model

Consider the linear system (1), where the coefficient matrix $\mathbf{A}$ is special matrix with elements $\mathbf{A} = (a_{i,j}), i, j = 1, 2, \ldots, n$, and $\mathbf{x}$,$\mathbf{b}$ are n-component unknown and known vectors given by $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, $\mathbf{b} = [b_1, b_2, \ldots, b_n]^T$. We rewrite formula (1) as follows.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b(1) \\ \qquad\qquad \vdots \\ a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n = b(k) \\ \qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b(n) \end{cases} \tag{2}$$

since,

$$\sum_{j=1}^{n} a_{kj}x_j^k = A(k,:)\mathbf{x}^k = b(k), \quad (k = 1, 2, \ldots, n) \tag{3}$$

Assumed that

$$y(k) = \mathbf{A}(k,:)\mathbf{x}^k \tag{4}$$

Where,$\mathbf{A}(k,:) = [a_{k1}, a_{k2}, \ldots, a_{kn}]$, $\mathbf{x}^k = [x_1^k, x_2^k, \ldots, x_n^k]^T$.

Suppose $a_{k1}, a_{k2}, \ldots, a_{kn}$ are the kth input samples of neural network,$x_1^k, x_2^k, \ldots, x_n^k$ are the weights of neural network, $y(k)$ and $b(k)$ is the output of neural network and the desired output of neural network at k^{th} moment respectively, then figure 1 shows the neural-network model.

3 The Neural-network Algorithm RLS-based

3.1 Algorithm description

The recursive least-square (RLS) algorithm has been extensively used in adaptive filtering, self-tuning control systems, interference cancellation, system identification [14-15]. Fig.1 depicts the

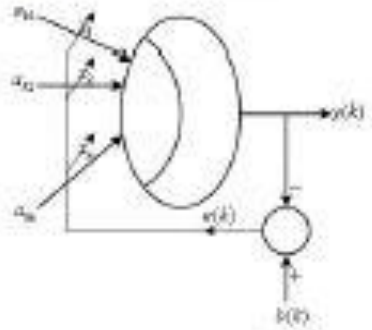

Fig. 1: The neural-network model

neural-network model for solving systems of problems (1). The conventional RLS algorithm for solving linear systems of (1) is given as following.

The output of the neural network:

$$y(k) = \sum_{j=1}^{n} a_{kj} x_j^k = A(k,:)\mathbf{x}^k \tag{5}$$

The error function:

$$e(k) = b(k) - y(k), \ \ (k = 1, 2, \ldots, n) \tag{6}$$

Define an object function:

$$J = 0.5 \sum_{k=1}^{n} e^2(k) \tag{7}$$

To minimize the objective function J, the weight vector $\mathbf{x}^k$ is recursively calculated via using a Recursive Least Squares (RLS) rule [14]. In the standard RLS algorithm, the update of the weight vector $\mathbf{x}^k$, the error signal $e(k)$, the Kalman gain vector Q^k, and the inverse of the correlation matrix P^k, are described as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{Q}^k e(k) \tag{8}$$

$$\mathbf{Q}^k = \frac{\mathbf{P}^k \mathbf{A}^T(k,:)}{\lambda + \mathbf{A}(k,:)\mathbf{P}^k \mathbf{A}^T(k,:)} \tag{9}$$

$$\mathbf{P}^{k+1} = \frac{1}{\lambda}[\mathbf{I} - \mathbf{Q}^k \mathbf{A}(k,:)]\mathbf{P}^k \tag{10}$$

here, $(k = 1, 2, \ldots, n)$,$\mathbf{P}^0 = \alpha \mathbf{I} \in \mathbf{R}^{n \times n}$ (generally, let $\alpha = 10^6 \sim 10^{16}$, $\mathbf{I} \in \mathbf{R}^{n \times n}$ is an identity matrix), and forgetting factor: $0.90 \leq \lambda \leq 1$.

3.2 Algorithm steps

Step1. Input the maximum iterative steps N and the stopping tolerance ε of neural-network method RLS-based, $J = 0$;

Step2. Input the initial weight vector $\mathbf{x}^0$, large positive real number α, and forgetting factor λ , let $n = length(b)$.

Step3. For $n = 1 : N$

For $k = 1 : n$

Compute: $e(k) = b(k) - y(k)$; $\mathbf{Q}^k = \frac{\mathbf{P}^k \mathbf{A}^T(k,:)}{\lambda + \mathbf{A}(k,:)\mathbf{P}^k \mathbf{A}^T(k,:)}$; $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{Q}^k e(k)$;

$\mathbf{P}^{k+1} = \frac{1}{\lambda}[\mathbf{I} - \mathbf{Q}^k \mathbf{A}(k,:)]\mathbf{P}^k$; $J = J + 0.5e^2(k)$;

End

If $J \leq \varepsilon$

Output $'x' = x, 'Iterations' = n$, break

Else

$J = 0$;

End

End

4 Numerical Examples

In this section, all numerical experiments are achieved in MATLAB. In actual computations, initial approximate value $\mathbf{x}^0$ is zero vector; and terminates once the current iteration step reaches N or the current iteration reaches a prescribed stopping tolerance ε, i.e. $J \leq \varepsilon$, and taking $N = 10$.

Example 1. Consider the integer Toeplitz system [6]:

$$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 1 & -1 \\ 2 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix} \tag{11}$$

Let $\alpha = 10^6$ and $\lambda = 1$. When the three weights of neural network have been trained for two iterations, the final solution and the property ..... respectively

$\mathbf{x} = \frac{1}{23}[1.7,9,34]^T$, which is the exact solution of Example 1, and $J = 6.1227\, e^{-32}$.

Example 2. Consider the Vandermonde system [6]:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 5 & 4 \\ 1 & 9 & 25 & 16 \\ 1 & 27 & 125 & 64 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 0 \end{bmatrix} \tag{12}$$

Let $\alpha = 10^{16}$ and $\lambda = 1$. When the four weights of neural network have been trained for three iterations, the final solution and the property index is respectively

$\mathbf{x} = \frac{1}{48}[-46, 108, 18, -80]^T$, which is the exact solution of Example 2, and $J = 8.46685e^{-29}$.

Example 3. Consider the Pascal system [11]:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 252 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} -2 \\ -6 \\ -8 \\ -4 \\ 11 \\ 43 \end{bmatrix} \tag{13}$$

The accuracy solution of the system (13) is $\mathbf{x}^* = [1, 0, -4, 0, 1, 0]^T$.

Let $\alpha = 10^{16}$ and $\lambda = 0.9$. When the six weights of neural network have been trained for seven iterations, the final solution and the property index is respectively

$\mathbf{x} = [1, 0, -4, 0, 1, 0]^T$, $J = 2.84976e^{-29}$, and $\|\mathbf{x}^* - \mathbf{x}\|_\infty = 1.957401e^{-14}$.

Example 4. Consider the Comrade system [12,13]:

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 3 & -2 & 1 & 0 & 0 \\ 0 & 1 & -3 & 1 & 0 \\ 0 & 0 & 2 & -4 & 1 \\ -1 & -1 & -1 & 3-1 & -5-1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -3 \\ -5 \\ -28 \end{bmatrix} \tag{14}$$

The exact solution of the system (14) is $\mathbf{x}^* = [1, 2, 3, 4, 5]^T$.

Let $\alpha = 10^{16}$ and $\lambda = 0.98$. When the five weights of neural network have been trained for three iterations, the final solution and the property index is respectively

$\mathbf{x} = [1, 2, 3, 4, 5]^T$, $J = 4.93038e^{-31}$, and $\|\mathbf{x}^* - \mathbf{x}\|_\infty = 2.22045e^{-16}$.

5 Concluding Remarks

In this paper, we have presented the new method (NN-RLS) for solving special linear system of equations, such as Toeplitz, Vandermonder, Pascal, and Comrade systems etc. Since there are efficient recursive least squares (RLS) for neural network, our algorithm may be useful. The results of four examples show that the NN-RLS method is very simple and effective.

Acknowledgement

This work is supported by the projects of National Natural Science Foundation of China (61040049), Natural Science Foundation of Hunan, China(11JJ6064), and Science and technology of Hunan, China(2011GK3122).

References

[1] B. Psenicka, F. Garcia-Ugalde, A. Herrera-Camacho. The bilinear Z transform by Pascal matrix and its application in the design of digital filters, IEEE Signal Process. Lett. 9(11)(2002), 368-370.

[2] Jacek Konopacki. The Frequency transformation by matrix operation and its application in IIR filters design, IEEE Signal Process. Lett. 12(1)(2005), 5-8.

[3] Alan Edelman, Gilbert Strang. Pascal matrices, http://web.mit.edu/18.06/www/Essays/pascalwork.pdf.

[4] V. Biolkova, D. Biolek. Generalized pascal matrix of first-order S-Z transforms, in: ICECS'99 Pafos, Cyprus 1999, pp. 929-931.

[5] S. Chandra Sekhara Rao, Sarita. A symmetric linear system solver, Appl. Math. Comput. 203(2008), 368-379.

[6] Miroslav Morháč. Error-free algorithms to solve special and general discrete systems of linear equations, Appl. Math. Comput. 202(2008), 1-23.

[7] Z. Zhi-zheng. The linear algebra of the generalized Pascal matrix, Linear Algebr. Appl. 283(1998), 289-299.

[8] M. Bayat, H. Teimoori. Pascal k-eliminated functional matrix and its property, Linear Algebr. Appl. 308(2000), 65-75.

[9] L. Aceto, D. Trigiante. The matrices of Pascal and other greats, Amer. Math. Monthly, 108(2001), 232-245.

[10] Zhi-Hui Tang, Ramani Duraiswami and Nail Gumerov. Fast algorithm to compute matrix-vector products for Pascal matrices, Technical Reports from UMIACS, 2004.

[11] X. G. Lv, T. Z. Huang and Z. G. Ren. A new algorithm for linear systems of the Pascal type, J. Compt. Appl. Math. 225(2009), 309-315.

[12] T. Sogabe. Numerical algorithms for solving comrade linear systems based on tridiagonal solvers, Appl. Math. Comput. 198(2008), 117-122.

[13] A. A. Karawia. Two algorithms for comrade linear systems, Appl. Math. Comput. 189(2007), 291-297.

[14] C. F. So, S. C. Ng, S. H. Leung. Gradient based variable forgetting factor RLS algorithm, Signal Processing, 83(2003), 1163-1175.

[15] Yegui Xiao, Liying Ma, Rabab Kreidieh Ward. Fast RLS Fourier analyzers capable of accommodating frequency mismatch, Signal Processing, 87(2007), 2197-2212.

[16] Musheng Deng, Zhezhao Zeng, Haibo Wu. Numerical Integration Approach Based on Recursive Least Square Algorithm, Journal of Computational Information Systems 6: 11(2010), 3695-3702.