International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:7, No:3, 2013

Spline basis neural network algorithm for numerical integration

Lina Yan, Jingjing Di and Ke Wang

Abstract—A new basis function neural network algorithm is proposed for numerical integration. The main idea is to construct neural network model based on spline basis functions, which is used to approximate the integrand by training neural network weights. The convergence theorem of the neural network algorithm, the theorem for numerical integration and one corollary are presented and proved. The numerical examples, compared with other methods, show that the algorithm is effective and has the characteristics such as high precision and the integrand not required known. Thus, the algorithm presented in this paper can be widely applied in many engineering fields.

Keywords—Numerical integration; Spline basis function; Neural network algorithm

I. Introduction

DEFINITE integral is very important in computational science and engineering. Under certain conditions, Newton-Leibniz formula $\int_a^b f(x) = F(b) - F(a)$, where $F'(x) = f(x)$ can be used to compute definite integrals. However, as is well known, the primitive of general integrand $f(x)$ is always hard to obtain or very complicated except a limited number of types. Moreover, in real problems, the function $f(x)$ is always presented as a function table without an analytic expression. Thus, the Newton-Leibniz formula fails to be used in most cases and we have to seek numerical methods for computing definite integrals.

By the definition of the definite integral, we can use the discretization or weighted average approximation of finite samples of integrand to substitute it, which is the main idea of numerical integration. With this idea, there are many methods of numerical integration, such as Newton-Cotes formula, Romberg's method and Gaussian quadrature[4], [8], [9]. Newton-Cotes formula is a common method by utilizing polynomial interpolation to construct numerical integration, but the convergence can't be guaranteed in high order case. Romberg's method has fast convergence and high precision while expensive computation cost. Gaussian quadrature also has fast convergence, high precision and is robust, but the computation of nodes and coefficients is complicated and the integrand $f(x)$ is required known. In 2004, Wang, He and Zeng[10] provided a method based on triangular basis neural network algorithm which is better than the traditional methods with higher precision, the integrand not required known and good to singular integral. The idea utilizing basis function is popular with scientists in signal processing, pattern recognition and artificial neural networks[2], [5], [11], [12], [13]. Spline function is easy to calculate and has many advantages as an interpolation function, thus, it's often used in numerical computations[3], [6], [7]. Given this, we presents a neural network algorithm with spline basis function that has higher precision than Wang, He and Zeng's, is also good for singular integral and does not require the integrand known too. Therefore, the method can solve the integral problems effectively for systems which are unknown or difficult to be modeled and has great applications in engineering practice.

Ke Wang is the corresponding author. E-mail: kwang@shu.edu.cn.
The authors are with the Department of Mathematics, College of Sciences, Shanghai University, Shanghai 200444, P.R. China. Yan's e-mail: linayan0718@126.com. Di's e-mail: 419287993@qq.com.

II. Neural network model based on spline basis function

Because spline function is simple in form, it can be easily computed. When the interpolation nodes are gradually increased, not only does it converge to the function itself, but also its derivative converges to the derivative of the function[15]. In this paper, the theory of spline approximation is adequately combined with the neural network principle by these advantages of spline function. As shown in Fig. 1, the

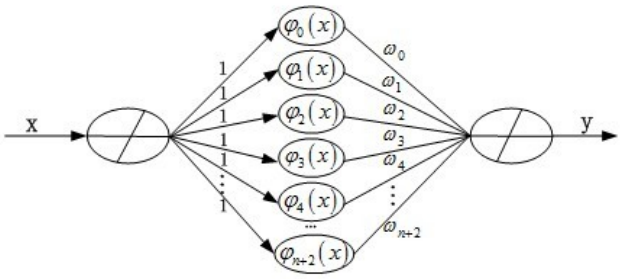

Fig. 1. Neural network model based on spline basis function

neural network model based on m times spline basis function is constructed, where ω_j is the weight of neural network and $\phi_j(x)$ is spline basis function. That is

$$\phi_j(x) = \begin{cases} x^j, & j = 0, 1, \cdots, m, \\ (x - x_{j-m})^m, & j = m+1, m+2, \cdots, N \end{cases} \tag{1}$$

is the hidden layer neuron incentive function, $x \in [0, 1]$. Suppose the weight matrix $W = [\omega_0, \omega_1, \cdots, \omega_N]^{\mathrm{T}}$ and the incentive matrix $\Phi(x) = [\phi_0(x), \phi_1(x), \cdots, \phi_N(x)]^{\mathrm{T}}$. So the output of the neural network is

$$y(x) = \sum_{j=0}^{N} \omega_j \phi_j(x) = W^{\mathrm{T}} \Phi(x). \tag{2}$$

The error function is

$$e(t) = f(x_k) - y(x_k), k = 0, 1, \cdots, n, \tag{3}$$

where $n+1$ is the training sample points, $n = N - m + 1$, $f(x)$ is the integrand, and the performance index

$$J = \frac{1}{2}\sum_{k=0}^{n} e^2(k). \tag{4}$$

The weight adjustment is

$$W(k+1) = W(k) + \eta e(k)\Phi(x_k), \tag{5}$$

where $0 < \eta < 1$ is the learning rate.

A. Convergence theorem of neural network model

Theorem 2.1: Let η be the learning rate. Then the neural network algorithm is convergent, when $0 < \eta \leq \frac{2}{\sum_{j=0}^{N} |\phi_j(x_k)|^2}$, where N is the number of hidden layer neurons.

Proof: Let the Lyapunov function $V(k) = \frac{1}{2}e^2(k)$. Then

$$\triangle V(k) = \frac{1}{2}e^2(k+1) - \frac{1}{2}e^2(k), \tag{6}$$

and $e(k+1) = e(k) + \triangle e(k) = e(k) + \left(\frac{\partial e(k)}{\partial W}\right)^T \frac{\partial e(k)}{\partial W}$, $\Delta W = -\eta e(k)\frac{\partial e(k)}{\partial W}$, so

$$\triangle e(k) = -\eta e(k)\left(\frac{\partial e(k)}{\partial W}\right)^T \frac{\partial e(k)}{\partial W} = -\eta e(k)\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2, \tag{7}$$

where $\|\cdot\|_2 = \sqrt{\sum |\cdot|^2}$ is the Euclidean norm. So formula (6) can be rewritten as

$$\begin{aligned}\triangle V(k) &= \frac{1}{2}[e(k) + \triangle e(k)]^2 - \frac{1}{2}e^2(k) \\ &= \triangle e(k)\left[e(k) + \frac{1}{2}\triangle e(k)\right] \\ &= -\eta e(k)\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2\left[e(k) - \frac{1}{2}\eta e(k)\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2\right] \\ &= \left\|\frac{\partial e(k)}{\partial W}\right\|_2^2 e(k)\left(-\eta + \frac{1}{2}\eta^2\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2\right).\end{aligned} \tag{8}$$

To make the neural network algorithm convergent, we have

$$-\eta + \frac{1}{2}\eta^2\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2 \leq 0, \eta > 0,$$

that is

$$0 < \eta \leq \frac{2}{\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2}. \tag{9}$$

From (2) and (3), we get

$$\frac{\partial e(k)}{\partial W} = \left(\frac{\partial e(k)}{\partial y(x_k)}\right)\left(\frac{\partial y(x_k)}{\partial W}\right) = -\Phi(x_k).$$

According to (1), it can be proved

$$\left\|\frac{\partial e(k)}{\partial W}\right\|_2^2 = \|-\Phi(x_k)\|_2^2 = \|\Phi(x_k)\|_2^2 = \sum_{j=0}^{N} |\phi_j(x_k)|^2. \tag{10}$$

When $0 < \eta \leq \frac{2}{\sum_{j=0}^{N} |\phi_j(x_k)|^2}$, we have $\triangle V(k) < 0$. Consequently, it is shown that the neural network algorithm is convergent. ■

B. Algorithm of neural network model

According to the neural network model based on the spline basis function which is discussed above, we get the following neural network algorithm for numerical integration:

1) *Take the learning rate satisfying Theorem 2.1, such as* $\eta = \frac{2}{N+1}$, *the training sample set* $\left\{x_k = a + \frac{b-a}{n}k | f(x_k), k = 0, 1, 2\cdots, n\right\}$ *and the initial weight matrix* $W_0 = \text{zeros}(N+1, 1)$*; Calculate the incentive matrix* $\Phi(x_k) = [\phi_0(x), \phi_1(x), \cdots, \phi_N(x)]^{\mathrm{T}}$*; Set the performance index* $J = 0$ *and give an infinitely small positive number* ε;
2) *Calculate the output of the neural network:* $y(x_k) = \sum_{j=0}^{N} \omega_j \phi_j(x_k) = W^{\mathrm{T}}\Phi(x_k)$;
3) *Calculate the error function:* $e(k) = f(x_k) - y(x_k)$;
4) *Calculate the performance index of the neural network:* $J = \frac{1}{2}\sum_{k=0}^{n} e^2(k)$;
5) *Adjust the neural network weight:* $W(k+1) = W(k) + \eta e(k)\Phi(x_k)$;
6) *If the sample set is not fully trained, return to step 2 and repeat the above steps; Otherwise, compare the performance index* J *and* ε*: If* $J > \varepsilon$*, set* $J = 0$*, return to step 2 and repeat the above steps; If* $J \leq \varepsilon$*, end training, output the neural network weights* $W = [\omega_0, \omega_1, \cdots, \omega_N]^{\mathrm{T}}$.

C. Theorem for numerical integration

After getting the neural network weights by the above algorithm, we can easily show the following numerical integration theorem and corollary.

Theorem 2.2: Let a, b be the upper and lower limits of the integral, $0 \leq a, b \leq 1$, and ω_j be the neural network weights. Then

$$I = \int_a^b f(x)\,\mathrm{d}x \approx \sum_{j=0}^{m} \frac{\omega_j(b^{j+1} - a^{j+1})}{j+1} \tag{11}$$

$$+ \sum_{j=1}^{n-1} \frac{\omega_{j+m}[(b - x_j)^{m+1} - (a - x_j)^{m+1}]}{m+1}. \tag{12}$$

Corollary 2.3: When $a = 0, b < 1$, we have

$$I = \int_0^b f(x)\,\mathrm{d}x \approx \sum_{j=0}^{m} \frac{\omega_j b^{j+1}}{j+1} \tag{13}$$

$$+ \sum_{j=1}^{n-1} \frac{\omega_{j+m}[(b - x_j)^{m+1} - (-x_j)^{m+1}]}{m+1}. \tag{14}$$

Remark 2.4: The integral interval is $[0, 1]$ for the numerical algorithm proposed in this paper, so we need to do some integral transform to make it fall in $[0, 1]$ if the integral interval is beyond this scope.

III. Numerical examples

In order to illustrate the validity and feasibility of the algorithm of this paper, some examples that mentioned in the references[1], [14] are selected and compared with the traditional integral methods. The numerical results show that the algorithm is effective and has the characteristics such as high precision and well-adapted. By the principle of interpolation, the low-order polynomials have less oscillation and better effectiveness than high-order ones to approximate known functions[15], so, in general, we take $m = 3$ or 4, and cubic spline function is selected in the following examples.

Example 1. Reference[1] use the trapezoidal formula and Simpson's method to calculate the integration of integrand $x^2, x^4, \sqrt{1+x^2}, \frac{1}{1+x}, \sin x, e^x$ in the interval [0,2], respectively. The results are shown in Table 4.7 of literature [1]. For our algorithm, we choose the neural network structure shown in Fig. 1 and select cubic spline basis function as the incentive function of neurons with the network structure $1 \times 120 \times 1$, the initial weight zero, learning rate $\eta = 0.0165$, the training sample set $\{x_k = \frac{1}{118}k | f(x_k), k = 0, 1, 2 \cdots, 118\}$, the stopping criteria is $J = 10^{-7}$ or iterations = 500, where $f(x)$ is integrand. The results of our algorithm and those in references[1], [10], [14] are listed in Table I.

From Table I, we can see that our algorithm has higher precision than others even triangular basis neural network algorithm.

TABLE I
THE COMPARISON OF NUMERICAL INTEGRATION METHODS

$f(x)$	x^2	x^4	$\sqrt{1+x^2}$	$\frac{1}{1+x}$	$\sin x$	e^x
Exact value	2.667	6.400	2.958	1.099	1.416	6.389
Trapezoidal	4.000	16.000	3.326	1.333	0.909	8.389
Simpson's	2.667	6.667	2.964	1.111	1.435	6.421
Triangular	2.665	6.393	2.959	1.101	1.415	6.388
Our results	2.667	6.402	2.958	1.098	1.416	6.390

Example 2. In order to show the algorithm has the ability to calculate singular integrals, we consider the following singular integral

$$f(x) = \begin{cases} e^{-x}, & 0 \le x < 1, \\ e^{-\frac{x}{2}}, & 1 \le x < 2, \\ e^{-\frac{x}{3}}, & 2 \le x < 3. \end{cases}$$

The accurate integral value is 1.5460. For our algorithm, take neural network hidden layer neurons $N = 120$, learning rate $\eta = 0.165$, and select the training sample set $\{x_k = \frac{1}{118}k | f(x_k), k = 0, 1, 2 \cdots, 118\}$, $J = 10^{-7}$. The integral result is 1.5462 (It's 1.5467 in [10]) and integral error curve is shown in Fig. 2.

We can see that our algorithm has higher precision than triangular basis neural network algorithm and has very good approximation effect for singular integral. Moreover, the convergence rate is quite fast.

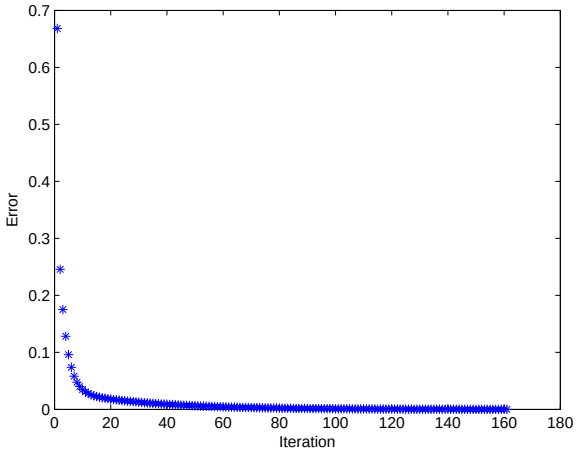

Fig. 2. Integral error curve

IV. Conclusion

We propose a basis function neural network algorithm for numerical integration. We construct the neural network model based on spline basis functions, which is used to approximate the integrand by training neural network weights. The convergence theorem of the neural network algorithm, the theorem for numerical integration and one corollary are presented and proved. The numerical examples, compared with other methods, show that the algorithm is effective and has the characteristics such as high precision and the integrand not required known. Thus, the algorithm presented in this paper can be widely applied in many engineering fields. Furthermore, we can discuss the numerical differentiation with the same idea, which will be the next work.

Acknowledgment

This work was supported by Shanghai College Teachers Visiting abroad for Advanced Study Program (B.60-A101-12-010) and Shanghai Leading Academic Discipline Project (J50101).

References

[1] R.L. Burden and J.D. Faires, Numerical Analysis(Seventh Edition), Brooks/Cole, Thomson Learning, Inc., 2001.
[2] F. Castillo, J. Arellano and S. Sánchez, Statistical approach to basis function truncation in digital interpolation filters, World Academy of Science, Engineering and Technology 39 (2009) 622-626.
[3] C. Dagnino, Product integration of singular integrands based on cubic spline interpolation at equally spaced nodes, Numerische Mathematik 57 (1990) 97-104.
[4] K. Deb, Multi-objective genetic algorithms: problem difficulties and construction of test problems, Evolutionary Computation 7(1999) 205-230.
[5] K. Elleuch and A. Chaari, Modeling and identification of hammerstein system by using triangular basis functions, World Academy of Science, Engineering and Technology 51 (2011) 1332-1336.
[6] S. Gao, Z. Zhang and C. Cao, Differentiation and numerical integral of the cubic spline interpolation, Journal of Computers 6 (2011) 2037-2044.
[7] Y. Isomoto, Numerical integration by bicubic spline function, Information processing in Japan 15 (1975) 16-20.

[8] J.H. Shen, Fundamentals of Numerical Calculation (in Chinese), Tongji University Press, Shanghai, 1999.
[9] N.C. Wang, A Concise Guide to Numerical Analysis (in Chinese), Higher Education Press, Beijing, 1997.
[10] X.-H. Wang, Y.-G. He and Z.-Z. Zeng, Numerical integration study based on triangle basis neural network algorithm (in Chinese), Journal of Electronics and Information Technology 26 (2004) 394-399.
[11] S. Yan, X. Chen, S. Dai and Q. Zhang, A kind of fast numerical integration method based on neural network algorithm, International Journal of Digital Content Technology and its Applications 6 (2012) 403-410.
[12] J. Yang and T. Du, Neural network algorithm for solving triple integral, 2010 Sixth International Conference on Natural Computation (ICNC 2010) 1 (2010) 412-416.
[13] Z.-Z. Zeng, Y.-N. Wang and H.Wen, Numerical integration based on a neural network algorithm, Computing in Science & Engineering 8 (2006) 42-48.
[14] Y.-Q. Zhou, M. Zhang and B. Zhao, Solving numerical integration based on evolution strategy method (in Chinese), Chinese Journal of Computers 31 (2008) 196-206.
[15] A.J. Zou and Y.N. Zhang, Basis Function Neural Networks and their Applications (in Chinese), Sun Yat-sen University Press, Guangzhou, 2009.