Math Sci (2014) 8:117
DOI 10.1007/s40096-014-0117-6

ORIGINAL RESEARCH

Utilizing artificial neural network approach for solving two-dimensional integral equations

B. Asady · F. Hakimzadegan · R. Nazarlue

Received: 22 February 2013 / Accepted: 5 March 2014 / Published online: 2 April 2014

Abstract This paper surveys the artificial neural networks approach. Researchers believe that these networks have the wide range of applicability, they can treat complicated problems as well. The work described here discusses an efficient computational method that can treat complicated problems. The paper intends to introduce an efficient computational method which can be applied to approximate solution of the linear two-dimensional Fredholm integral equation of the second kind. For this aim, a perceptron model based on artificial neural networks is introduced. At first, the unknown bivariate function is replaced by a multilayer perceptron neural net and also a cost function to be minimized is defined. Then a famous learning technique, namely, the steepest descent method, is employed to adjust the parameters (the weights and biases) to optimize their behavior. The article also examines application of the method which turns to be so accurate and efficient. It concludes with a survey of an example in order to investigate the accuracy of the proposed method.

Keywords Two-dimensional integral equations · Neural networks · Learning algorithm · Cost function · Approximate solution

Introduction

Recently, integral equations have been extensively investigated theoretically and numerically. Note that they occur in a wide variety of physical applications, various fields of neural sciences and numerous applications such as electrical engineering, economics, elastically, plasticity, etc. Since these equations usually cannot be solved explicitly, it is going to be obtained in approximate solutions. There are several numerical methods for approximating solution of Fredholm and Volterra integral equations in one- and two-dimensions. For example, Tricomi in his book [25] introduced the classical method of successive approximations for integral equations. Variational iteration method [15] was effective and convenient for solving integral equations. The Homotopy analysis method (HAM) was proposed by Liao [16] and then has been applied in [1]. The Taylor expansion approach was presented for solving integral equations by Kanwal and Liu [14] and then has been extended in [17]. In addition, Jafari et al. [12] applied Legendre wavelets method to find numerical solution of linear integral equations. In [13] an architecture of artificial neural networks (NNs) was suggested to approximate solution of linear Fredholm integral equations systems. For this aim, first the truncation of the Taylor expansions for unknown functions was substituted in the origin system. Then the purposed neural network has been applied for adjusting the real coefficients of given expansions in resulting system. In [9], a numerical method based on feed-forward neural networks has been presented for solving Fredholm integral equations of the second kind. The Bernstein polynomials have frequently been applied in the solution of integral equations and approximation theory [5–7, 19, 20]. Also, there are many articles which deal with the solution and analysis of two-dimensional Fredholm and Volterra integral equations. Mirzaei and Dehghan [22] described a numerical scheme based on the moving least squares (MLS) method for solving integral equations in one- and two-dimensional spaces. The method was a meshless method, since it did not require any background

B. Asady (✉) · F. Hakimzadegan · R. Nazarlue
Department of Mathematics, Islamic Azad University, Science and Branch, Arāk, Iran
e-mail: babakmz2002@yahoo.com; b-asadi@iau-arak.ac.ir

interpolation or approximation cells and it did not depend on the geometry of domain. Hadizadeh and Asgary [11] using the bivariate Chebyshev collocation method solved the linear Volterra–Fredholm integral equations of the second kind. Alipanah and Esmaeili [2] approximated the solution of the two-dimensional Fredholm integral equation using Gaussian radial basis function based on Legendre–Gauss–Lobatto nodes and weights. Two-dimensional orthogonal triangular functions are used in [3, 18] as a new set of basis functions to approximate solutions of nonlinear two-dimensional integral equations. Babolian et al. [4] applied two-dimensional rationalized Haar functions for finding the numerical solution of nonlinear second kind two-dimensional integral equations. They reduced the present problem to solve a nonlinear system of algebraic equations using bivariate collocation method and Newton–Cotes nodes. Moreover, some different valid methods for solving these kind of equations have been developed.

This paper focuses on constructing a new algorithm with the use of feed-forward neural networks to reach an approximate solution of the linear two-dimensional Fredholm integral equation. For this purpose, first unknown two-variable function in the problem is replaced by a three-layer perceptron neural network. Supposedly, the limits of integrations are partitioned into set points, this architecture of neural networks can calculate the output corresponding to input vector. Now a cost function to be minimized is defined on the set points. Consequently, the suggested neural net using a learning algorithm that is based on the gradient descent method adjusts parameters (the weights and biases) to any desired degree of accuracy. Here is an outline of the paper. In "Preliminaries", the basic notations and definitions of the integral equations and the artificial neural networks are briefly presented. "The general method" describes how to find approximate solution of the given two-dimensional integral equations using proposed approach. Finally in "An example", an numerical example is provided and results are compared with the analytical solutions to demonstrate the validity and applicability of the method.

Preliminaries

In this section we will focus on the basic definitions and introductory concepts in integral equations. In addition the basic principles of artificial neural network (ANN) approach are presented and reviewed for solving linear second kind two-dimensional integral equations (2D-IEs).

Integral equations

Integral equations appear in many scientific and engineering applications, especially when initial value problems for boundary value problems are converted to integral equations. As stated before, we will review some integral equations and linear two-dimensional integral equations of the second kind as well.

Definition 2.1 Let $f : [a, b] \to \mathbb{R}$. For each partition $P = \{t_0, t_1, \ldots, t_n\}$ of $[a, b]$ and for arbitrary $\xi_i \in [t_{i-1}, t_i]$ $(1 \le i \le n)$, suppose

$$R_P = \sum_{i=1}^{n} f(\xi_i)(t_i - t_{i-1}),$$

$$\Delta := max\{|t_i - t_{i-1}|,\ i = 1, \ldots, n\}.$$

The definite integral of $f(t)$ over $[a, b]$ is

$$\int_a^b f(t)\mathrm{d}t = \lim_{\Delta \to 0} R_P$$

provided that this limit exists in the metric D [25].

Definition 2.2 The linear two-dimensional Fredholm integral equation (2D-FIE) of the second kind is presented by the form [2]

$$F(x,y) = f(x,y) + \lambda \int_c^d \int_a^b k(x,y,s,t)F(s,t)\mathrm{d}s\mathrm{d}t, \quad (x,y) \in [a,b] \times [c,d] \tag{1}$$

where λ is a constant parameter, the kernel k and f are given analytic functions on $L^2([a, b] \times [c, d])$. The two-variable unknown function F that must be determined appears inside and outside the integral signs. This is a characteristic feature of a second kind integral equation. It is important to point out that if the unknown function appears only inside the integral signs, the resulting equation is of first kind.

If the kernel function satisfies $k(x, y, s, t) = 0$, $s > x$, $t > y$ in Eq. (1), we obtain the linear two-dimensional Volterra integral equation (2D-VIE) [24]

$$F(x,y) = f(x,y) + \lambda \int_c^y \int_a^x k(x,y,s,t)F(s,t)\mathrm{d}s\mathrm{d}t, \quad (x,y) \in [a,b] \times [c,d]. \tag{2}$$

It should be noted that, if one of the limits of integration varies, the integral equation is called a Volterra–Fredholm integral equation. It is clear that, two-dimensional integral equations appear in many forms. Three distinct ways that depend on the limits of integration are used to characterize these equations which have been are briefly introduced. Notice that, if the function $f(x, y)$ in the present integral equations is identically zero, the equation is called homogeneous. Otherwise it is called inhomogeneous. These three concepts play a major role in the structure of the solution.

Artificial neural networks

Artificial neural networks (ANNs) can be considered as simplified computational structures that are inspired by observed process in natural networks of biological neurons in the brain. They are nonlinear mapping architectures based on the function of the human brain, therefore can be considered as powerful tools for modeling, especially when the underlying data relationship is unknown. A very important feature of these networks is their adaptive nature, where "learning by example" replaces "programming" in solving problems. In other words, in contrast to conventional methods, which are used to perform specific task, most neural networks are more versatile. This feature raises a very appealing computational model which can be applied to solve variety of problems.

The multilayer feed-forward neural network or multilayer perceptron (MLP) that had been proposed by Rosenblatt [23] is very popular and is used more than other neural network type for a wide variety of tasks. The present network learned by back-propagation algorithm is based on supervised procedure. In other words, the network constructs a model based on examples of data with known output.

In this subsection, an architecture of MLP model is discussed here briefly. We intend to give a short review on learning of the given neural network. First consider a three-layer ANN with two input units, N neurons in hidden layer and one output unit. Mathematical representation of the present neural network is given in Fig. 1. Using the figure, input–output relation of each unit and calculated output $u_N(x, y)$ can be written as follows:

Input units:

The input neurons make no change in their inputs, so:

$$o_1 = x, \tag{3}$$

$$o_2 = y.$$

Hidden units:

Input into a node in hidden layer is a weighted sum of outputs from nodes connected to it. Each unit takes its net input and applies an activation function to it. The input/output relation is normally given as follows:

$$O_p = g(net(p)), \tag{4}$$

$$net(p) = \sum_{i=1}^{2} (w_{pi}.o_i) + b_p, \quad p = 1, \ldots, N.$$

where $net(p)$ describes the result of the net outputs o_i impacting on unit p. Also, w_{pi} are weights connecting neuron i to neuron p and b_p is a bias for neuron p. Bias term is baseline input to a node in absence of any other inputs.

Output unit:

$$u_N(x, y) = \sum_{p=1}^{N} net(p). \tag{5}$$

The general method

In this section, we intend to use the MLP method to get a new numerical approach for solving the linear two-dimensional Fredholm integral equation of the second kind. In other words, how to apply this method to make a series approximation for the solution $F(x, y)$ in (1) will be

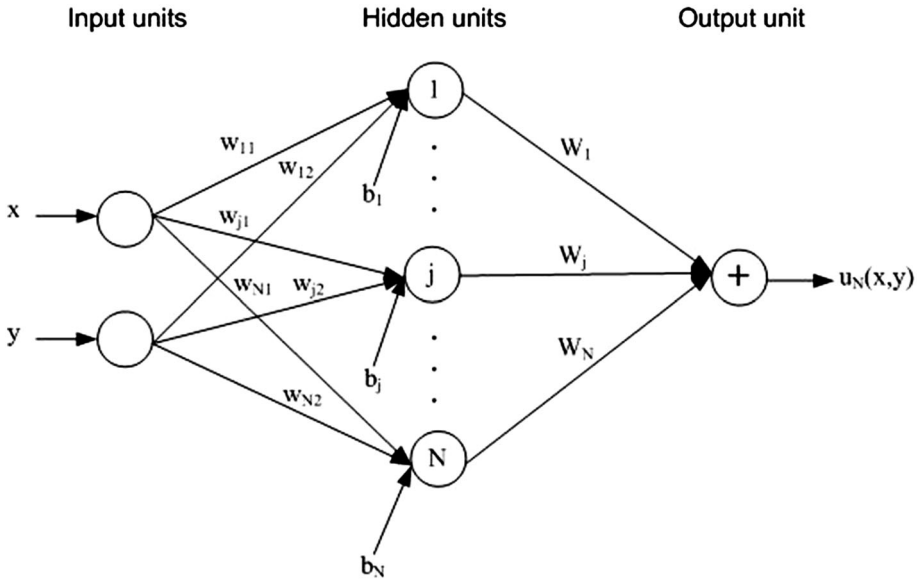

Fig. 1 Schematic diagram of the proposed MLP

described. The output of two-layer MLP network that is defined in Eq. (1) can be rewritten as follows:

$$u_N(x,y) = \sum_{p=1}^{N} \sum_{i=1}^{2} g(w_{pi}.o_i + b_p). \tag{6}$$

In order to approximate function u, first the intervals $[a\,b]$ and $[c\,d]$ are partitioned into set points x_i and y_j, respectively. Thus, the following set of equations will be obtained:

$$u_N(x_i,y_j) = \sum_{p=1}^{N} g\big(w_{p1}.x_i + w_{p2}.y_j + b_p\big). \tag{7}$$

Cost function

First suppose that $u_N(x,y)$ is the approximate solution with the adjustable parameters (wights and biases) for the unknown $F(x,y)$. After substituting this solution instead of the unknown function in the given 2D-FIE, the Eq. (1) can be transformed to a sum squared error minimization problem corresponding to the proposed neural network. So, the error function is regarded as a function on the weights and biases space of the net for $x = x_i$ and $y = y_j$ as follows:

$$E^{i,j}(w,W,b) := \frac{1}{2}\left(E_N^{i,j}(w,W,b)\right)^2, \tag{8}$$

where

$$E_N^{i,j}(w,W,b) = u_N(x_i,y_j) - f(x_i,y_j) - \lambda \int_c^d \int_a^b k(x_i,y_j,s,t) u_N(s,t)\mathrm{d}s\mathrm{d}t.$$

Now the total error of the network is defined as:

$$E(w,W,b) = \sum_{i,j} E^{i,j}(w,W,b). \tag{9}$$

The goal then is to minimize this function; therefore, we must deduce a back-propagation learning algorithm using the present cost function.

Proposed learning algorithm

Multilayer feed-forward neural network is learned by back-propagation algorithm that is based on supervised procedure. In other words, the MLP network is trained using a supervised learning algorithm which uses the training data to adjust the network weights and biases. Now let $w_{p,q}$, W_p and b_p (for $p = 1,\ldots,N$; $q = 1,2$) are initialized at small random values for input signals. For parameter $w_{p,q}$ adjustment rule can be written as follows:

$$w_{p,q}(r+1) = w_{p,q}(r) + \Delta w_{p,q}(r),\ p = 1,\ldots,N;\ q = 1,2, \tag{10}$$

$$\Delta w_{p,q}(r) = -\eta.\frac{\partial E^{i,j}}{\partial w_{p,q}} + \alpha.\Delta w_{p,q}(r-1), \tag{11}$$

where r is the number of adjustments, η is the learning rate and α is the momentum term constant. Similarly this adjustment rule can be written for other weight parameters. Thus, our problem is to calculate the derivative $\frac{\partial E^{i,j}}{\partial w_{p,q}}$ in (11). The derivative can be calculated as follows:

$$\frac{\partial E^{i,j}}{\partial w_{p,q}} = \frac{\partial E^{i,j}}{\partial E_N^{i,j}} \cdot \frac{\partial E_N^{i,j}}{\partial w_{p,q}}, \tag{12}$$

where

$$\frac{\partial u_N(x_i,y_j)}{\partial w_{p,q}} = \left(\frac{\partial u_N(x_i,y_j)}{\partial O(p)} \cdot \frac{\partial O(p)}{\partial net(p)} \cdot \frac{\partial net(p)}{\partial w_{p,q}}\right).$$

Consequently,

$$\frac{\partial E^{i,j}}{\partial w_{p,q}} = \tag{13}$$

$$\begin{cases} E_N^{i,j}.W_p.(x_i.g'(net(p)) - \lambda \int_c^d \int_a^b s.k(x_i,y_j,s,t).g'(w_{p,1}s + w_{p,2}t + b_p)\mathrm{d}s\mathrm{d}t), & \text{q=1} \\ E_N^{i,j}.W_p.(y_j.g'(net(p)) - \lambda \int_c^d \int_a^b t.k(x_i,y_j,s,t).g'(w_{p,1}s + w_{p,2}t + b_p)\mathrm{d}s\mathrm{d}t), & \text{q=2} \end{cases}.$$

Using a similar procedure as mentioned above, we have the correspondingly corollary for parameters W_p and b_p, in which we are refrained from going through proof details. So, we have:

$$\frac{\partial E^{i,j}}{\partial W_p} = E_N^{i,j}.(net(p) - \lambda \int_c^d \int_a^b k(x_i,y_j,s,t).g(w_{p,1}s + w_{p,2}t + b_p)\mathrm{d}s\mathrm{d}t), \tag{14}$$

and

$$\frac{\partial E^{i,j}}{\partial b_p} = E_N^{i,j}.W_p.(g'(net(p)) - \lambda \int_c^d \int_a^b k(x_i,y_j,s,t). \times g'(w_{p,1}s + w_{p,2}t + b_p)\mathrm{d}s\mathrm{d}t). \tag{15}$$

The MLP neural nets are the sample of regular networks, therefore they can approximate any continuous function on a compact set to arbitrary accuracy [10]. Now the learning algorithm can be summarized as follows:

Learning process

Step 1: $\eta > 0$, $\alpha > 0$ and $Emax > 0$ are chosen. Then quantities $w_{p,q}$, W_p *and* b_p $(p = 1,\ldots,N;\ q = 1,2)$ are initialized at small random values.

Step 2: Let $r := 0$ where r is the number of iterations of the learning algorithm. Then the running error E is set to 0.

Step 3: Let $r := r + 1$. Repeat below procedure for different values of i and j:

i Forward calculation: Calculate the output vector $u_N(x_i, y_j)$ by presenting the input vectors x_i and y_j.
ii Back propagation: Adjust the parameters $w_{p,q}$, W_p *and* b_p using the cost function (8).

Step 4: Cumulative cycle error is computed by adding the present error to E.
Step 5: The training cycle is completed. For $E < Emax$ terminate the training session. If $E > Emax$ then E is set to 0 and we initiate a new training cycle by going back to *Step 3*.

An example

In this section, in order to investigate the accuracy of the proposed method, we have chosen an example of linear two-dimensional integral equations of the second kind. For the example, the computed values of the approximate solution are calculated over a number of iterations and the cost function is plotted. Also, to show the efficiency of the present method for our problem, results will be compared with the exact solution.

Example 4.1 Consider the linear 2D-FIE

$$F(x,y) = f(x,y) + \int_0^1 \int_0^1 (s.sin(t) + 1)F(s,t)\mathrm{d}s\mathrm{d}t, \qquad (16)$$

where

$$f(x,y) = x.cos(y) - \frac{1}{6}\, sin(1)(3 + sin(1)),$$

with the exact solution $F(x,y) = x.cos(y)$. In this example, we illustrate the use of the FNN technique to approximate the solution of this integral equation. In the following simulations, we use the specifications as follows:

1. The number of hidden units: $N = 3$,
2. Learning rate $\eta = 0.5$,
3. Momentum constant $\alpha = 0.05$.

Numerical result can be found in Table 1, and Fig. 2 shows the cost function in the 20 iterations. Figures 3, 4, 5, 6 show the convergence behaviors for computed values of the weight parameters $w_{p,q}$ and W_p, bias b_p for different number of iterations.

There is no magic formula for selecting the optimum number of hidden neurons. However, some thumb rules are available for calculating number of hidden neurons. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters [21]. For a three-layer network with n input and m output neurons, the hidden layer would have at least $[\sqrt{nm}] + 1$ neurons.

To show convergence of the proposed method we solve Example 4.1 using shifted Legandre collocation method.

Table 1 Numerical results for example 4.1 by FNN technique

(x, y) = (0.1r, 0.1r)	Exact solution	Approximate solution		Error	
		$N = 3$	$N = 8$	$N = 3$	$N = 8$
$r = 1$	0.09950	0.09823	0.09905	0.00127	0.000444
$r = 2$	0.19601	0.19445	0.19585	0.00155	0.000143
$r = 3$	0.28660	0.28657	0.28654	0.00073	0.000056
$r = 4$	0.36842	0.36810	0.36836	0.00030	0.000037
$r = 5$	0.43879	0.43857	0.43867	0.00013	0.000023
$r = 6$	0.49520	0.49515	0.49518	0.00005	0.000013
$r = 7$	0.53539	0.53525	0.53528	0.00005	0.000013
$r = 8$	0.55737	0.55725	0.55728	0.00005	0.000013
$r = 9$	0.55945	0.55935	0.55938	0.00005	0.000013

The reason for choosing shifted Legandre collocation method is its simplicity. The details of shifted Legandre collocation method are as follows.

Shifted Legandre collocation method

The Legendre polynomials, $P_n(x), n = 0, 1, \ldots$, are the eigenfunctions of the singular Sturm–Liouville problem

$$\big((1 - x^2)P_n'(x)\big)' + n(n+1)P_n(x) = 0.$$

Also, they are orthogonal with respect to L^2 inner product on the interval $[-1, 1]$ with the weight function $w(x) = 1$, that is

$$\int_{-1}^{1} P_n(x)P_m(x)\mathrm{d}x = \frac{2}{2n+1}\delta_{nm},$$

where δ_{nm} is the Kronecker delta. The Legendre polynomials satisfy the recursion relation

$$P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x),$$

where $P_0(x) = 1$ and $P_1(x) = x$. If $P_n(x)$ is normalized so that $P_n(1) = 1$, then for any n, the Legendre polynomials in terms of power of x are

$$P_n(x) = \frac{1}{2^n}\sum_{m=0}^{\left[\frac{n}{2}\right]}(-1)^m \binom{n}{m}\binom{2n-2m}{n}x^{n-2m},$$

where $\left[\frac{n}{2}\right]$ denotes the integer part of $\frac{n}{2}$.

The Legendre–Gauss–Lobatto (LGL) collocation points $-1 = x_0 < x_1 < \cdots < x_N = 1$ are the roots of $P_N'(x)$ together with the points -1 and 1. Explicit formulas for the LGL points are not known. The LGL points have the property that

$$\int_{-1}^{1} p(x)\mathrm{d}x = \sum_{i=0}^{N} w_i p(x_i),$$

which is exact for polynomials of degree at most $2N - 1$, where w_i, $0 \leqslant i \leqslant N$, are LGL quadrature weights. For more details about Legendre polynomials, see [8].

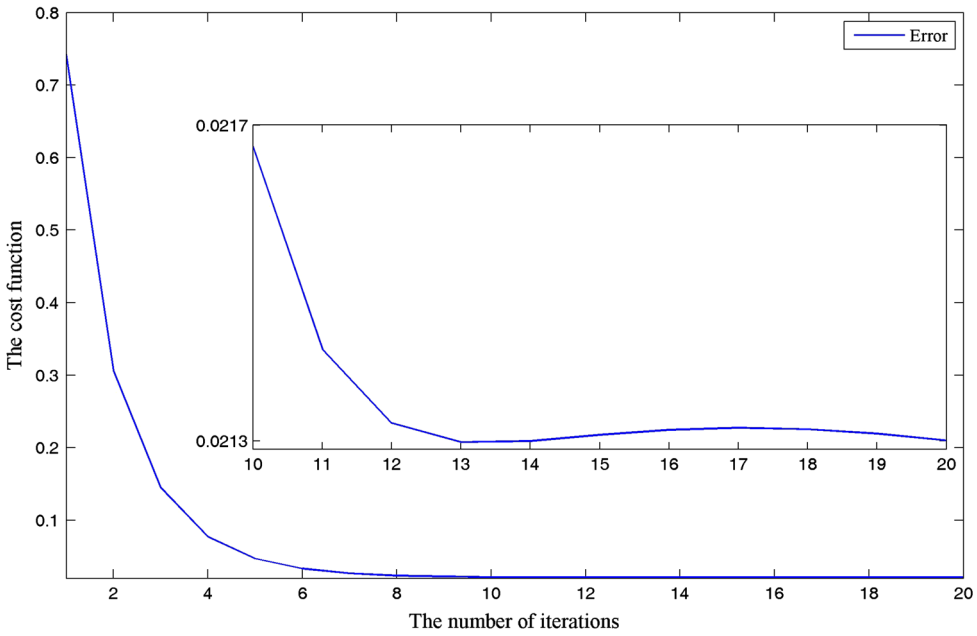

Fig. 2 The cost function for Example 4.1 on the number of iterations

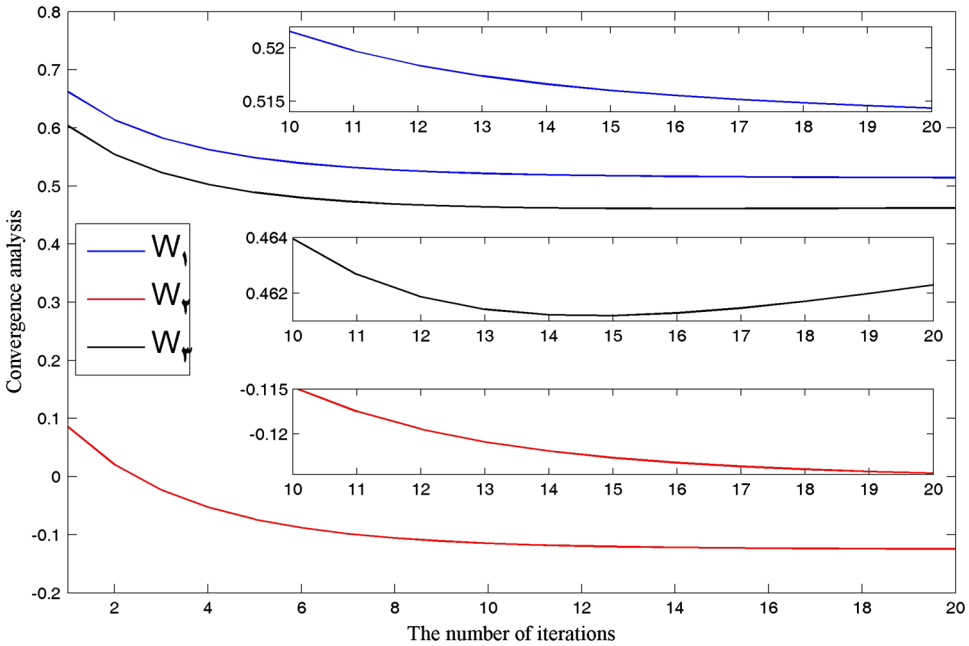

Fig. 3 Convergence of the weights W_r for Example 4.1

The shifted Legendre polynomials (ShLP) on the interval $t \in [0, 1]$ are defined by

$$\widehat{P}_n(t) = P_n(2t - 1), \quad n = 0, 1, \ldots,$$

which are obtained by an affine transformation from the Legendre polynomials. The set of ShLP is a complete $L^2[0, 1]$-orthogonal system with the weight function $w(t) = 1$. Thus, any function $f \in L^2[0, 1]$ can be expanded in terms of ShLP.

The ShLGL (Shifted Legendre–Gauss–Lobatto) collocation points $0 = t_0 < t_1 < \cdots < t_N = 1$ on the interval $[0, 1]$ are obtained by shifting the LGL points, x_i, using the transformation

$$t_i = \frac{1}{2}(x_i + 1), \quad i = 0, 1, \ldots, N. \tag{17}$$

Thanks to the property of the standard LGL quadrature, it follows that for any polynomial p of degree at most $2N - 1$ on $(0, 1)$,

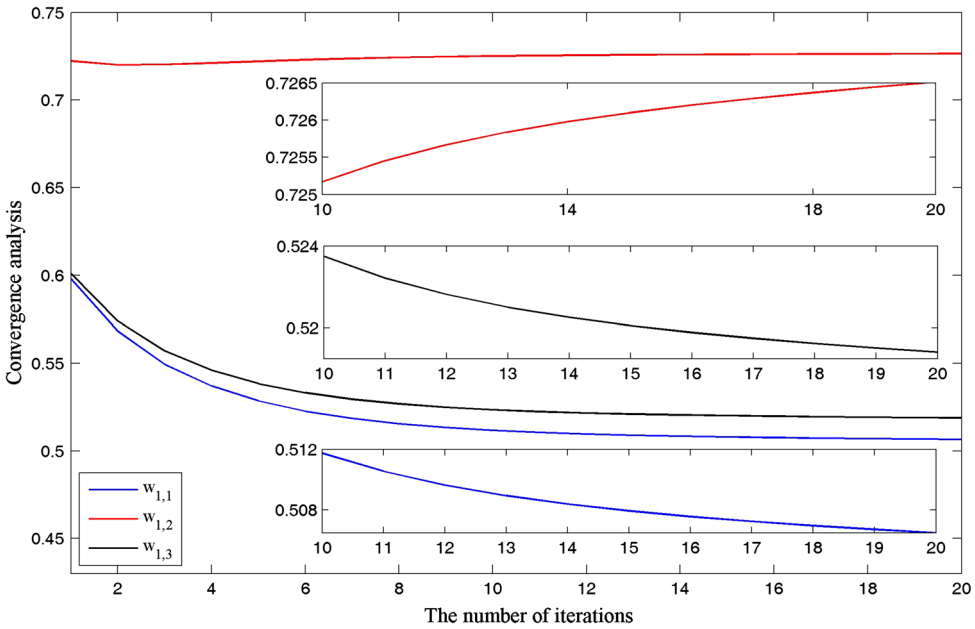

Fig. 4 Convergence of the weights $w_{1,r}$ for Example 4.1

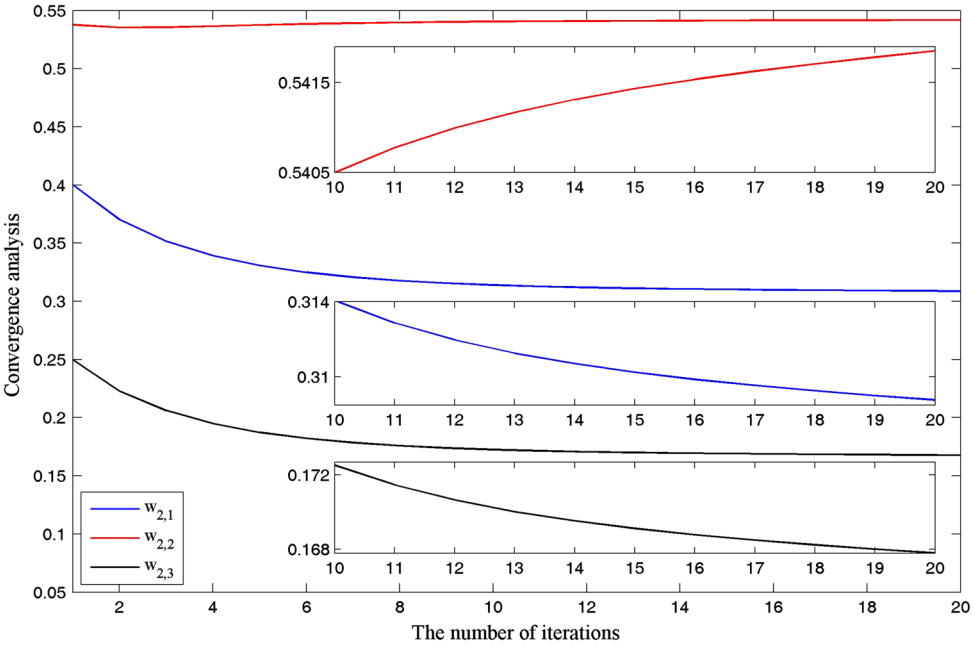

Fig. 5 Convergence of the weights $w_{2,r}$ for Example 4.1

$$\int_0^1 p(t)\mathrm{d}t = \frac{1}{2}\int_{-1}^{1} p\left(\frac{1}{2}(x+1)\right)\mathrm{d}x = \frac{1}{2}\sum_{i=0}^{N} w_i p\left(\frac{1}{2}(x_i+1)\right) = \sum_{i=0}^{N} \hat{w}_i p(t_i),$$

where $\hat{w}_i = \frac{1}{2}w_i$, $0 \leqslant i \leqslant N$, are ShLGL quadrature weights. The results stated above are also satisfied for Legendre–Gauss and Legendre–Gauss–Radau quadrature rules.

The function $F(x,y)$ is approximated by a ShLP of degree at most N as

$$F(x,y) = \sum_{i=0}^{N}\sum_{j=0}^{N} \alpha_{ij}\widehat{P}_i(x)\widehat{P}_j(y) \tag{18}$$

Now, by substituting (18) and collocation points (17) in (16), we have

$$\sum_{i=0}^{N}\sum_{j=0}^{N} \alpha_{ij}\widehat{P}_i(t_k)\widehat{P}_j(t_l)\bar{f}(t_k,t_l) + \int_0^1\int_0^1 (s.sin(t)+1) \sum_{i=0}^{N}\sum_{j=0}^{N} \alpha_{ij}\widehat{P}_i(s)\widehat{P}_j(t)\mathrm{d}s\mathrm{d}t, \quad k,l = 0,1,\ldots,N$$

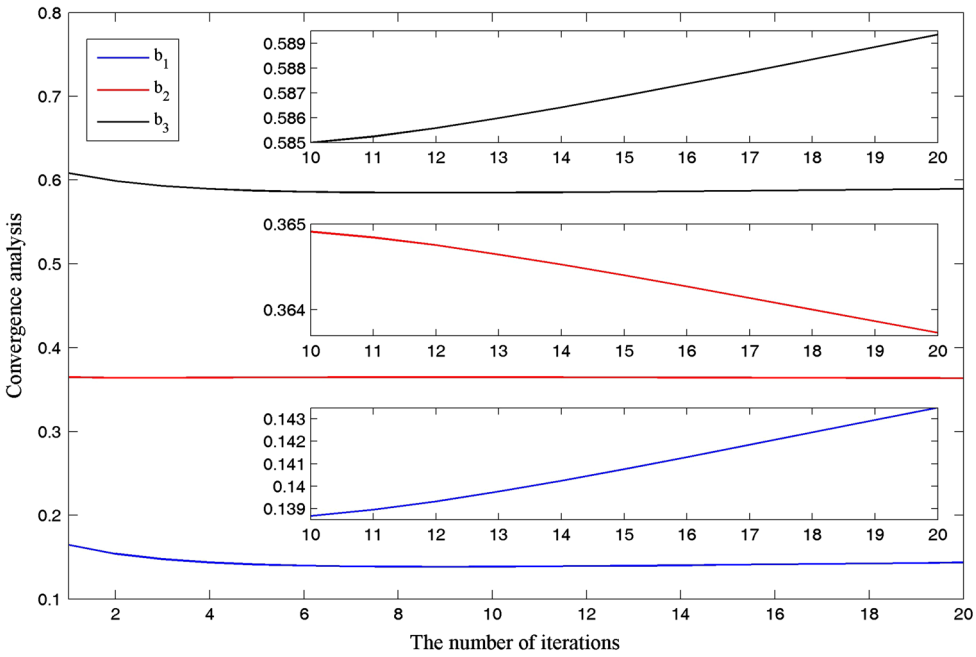

Fig. 6 Convergence of the weights b_r for Example 4.1

Table 2 Numerical results for Example 4.1 by SHLGL

(x, y) = (0.1r, 0.1r)	Exact solution	Approximate solution		Error	
		$N = 2$	$N = 3$	$N = 2$	$N = 3$
$r = 1$	0.09950	0.10497	0.09951	0.00547	0.00001
$r = 2$	0.19601	0.19855	0.19575	0.00254	0.00026
$r = 3$	0.28660	0.28268	0.28610	0.00392	0.00050
$r = 4$	0.36842	0.35734	0.36798	0.01108	0.00044
$r = 5$	0.43879	0.42256	0.43878	0.01623	0.00001
$r = 6$	0.49520	0.47832	0.49589	0.01688	0.00069
$r = 7$	0.53539	0.52462	0.53671	0.01077	0.00132
$r = 8$	0.55737	0.56147	0.55865	0.00410	0.00128
$r = 9$	0.55945	0.58885	0.55910	0.02940	0.00035

By solving this linear system we can find α_{ij}, $i,j = 0, 1, \ldots, N$, and then approximate the solution $F(x, y)$.

We solved the Example 4.1 using the method described for $N = 2$ and $N = 3$. Results are shown in Table 2. By comparing Tables 1 and 2 we find that obtained results in Table 1 are in concordance.

Conclusions

This paper suggested a new computational method to solve a two-dimensional Fredholm integral equations. So, a feed-forward artificial neural network has been proposed. This network is able of estimating approximate solution of assumed equation using the learning algorithm which is based on steepest descent rule. Clearly, in order to obtain accurate solution, many learning procedure should be considered. The analyzed examples illustrated the ability and reliability of the present approach. The obtained solutions, in comparison with exact solutions admit a remarkable accuracy. Extensions to the case of more general of integral equations are left for future studies.

References

1. Abbasbandy, S.: Numerical solution of integral equation: Homotopy perturbation method and Adomians decomposition method. Appl. Math. Comput. **173**, 493–500 (2006)
2. Alipanah, A., Esmaeili, Sh: Numerical solution of the two-dimensional Fredholm integral equations using Gaussian radial basis function. J. Comput. Appl. Math. **235**, 5342–5347 (2011)
3. Babolian, E., Maleknejad, K., Roodaki, M., Almasieh, H.: Two-dimensional triangular functions and their applications to nonlinear 2D Volterra-Fredholm integral equations. Comput. Math. Appl. **60**, 1711–1722 (2010)
4. Babolian, E., Bazm, S., Lima, P.: Numerical solution of nonlinear two-dimensional integral equations using rationalized Haar functions. Commun. Nonlinear Sci. Numer. Simulat. **16**, 1164-1175 (2011)
5. Bhatta, D.D., Bhatti, M.I.: Numerical solution of KdV equation using modied Bernsein polynomials. Appl. Math. Comput. **174**, 1255–1268 (2006)
6. Bhatti, M.I., Bracken, P.: Solutions of dierential equations in a Bernstein polynomial basis. J. Comput. Appl. Math. (2007). doi:10.1016/j.cam2006.05.002

7. Bhattacharya, S., Mandal, B.N.: Use of Bernstein polynomials in numerical solution of Volterra integral equations. Appl. Math. Sci. **36**(2), 1773–1787 (2008)
8. Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A.: Spectral methods: fundamentals in single domains. Springer, Berlin (2006)
9. Effati, S., Buzhabadi, R.: A neural network approach for solving Fredholm integral equations of the second kind. Neural Comput. Appl. doi:10.1007/s00521-010-0489-y
10. Fuller, R.: Neural Fuzzy Systems. Abo Akademi University, Department of Information Thechnologies (1995)
11. Hadizadeh, M., Asgary, M.: An ecient numerical approximation for the linear class of mixed integral equations. Appl. Math. Comput. **167**, 1090–1100 (2005)
12. Jafari, H., Hosseinzadeh, H., Mohamadzadeh, S.: Numerical solution of system of linear integral equations by using Legendre wavelets. Int. J.Open Prob. Comput. Math. **5**, 63–71 (2010)
13. Jafarian, A., Measoomy Nia, S.: Utilizing feed-back neural network approach for solving linear Fredholm integral equations system. Appl. Math. Mode. (2012). doi:10.1016/j.apm
14. Kanwal, R.P., Liu, K.C.: A Taylor expansion approach for solving integral equations. Int. J. Math. Educ. Sci. Technol. **20**, 411–414 (1989)
15. Lan, X.: Variational iteration method for solving integral equations. Comput. Math. Appl. **54**, 1071–1078 (2007)
16. Liao, S.J.: Beyond perturbation: introduction to the homotopy analysis method. Chapman Hall/CRC Press, Boca Raton (2003)
17. Maleknejad, K., Aghazadeh, N.: Numerical solution of Volterra integral equations of the second kind with convolution kernel by using Taylor-series expansion method. Appl. Math. Comput. **161**, 915–922 (2005)
18. Maleknejad, K., Jafari Behbahani, Z.: Applications of two-dimensional triangular functions for solving nonlinear class of mixed Volterra-Fredholm integral equations. Math. Comput. Mode (2011). doi:10.1016/j.mcm.2011.11.041
19. Maleknejad, K., Basirat, B., Hashemizadeh, E.: A Bernstein operational matrix approach for solving a system of high order linear Volterra–Fredholm integro-differential equations. Math. Comput. Mode **55**, 1363–1372 (2012)
20. Mandal, B.N., Bhattacharya, S.: Numerical solution of some classes of integral equations using Bernstein polynomials. Appl. Math. Comput. **190**, 1707–1716 (2007)
21. Masters, T.: Practical neural network recipes in C++. Academic press, NewYork (1993)
22. Mirzaei, D., Dehghan, M.: A meshless based method for solution of integral equations. Appl. Numer. Math. **60**, 245–262 (2010)
23. Rosenblatt, F.: The perceptron: a probabilistic model for information storage ang organization in the brain. Psychol. Rev. **65**, 386–408 (1958)
24. Tari, A., Rahimib, M.Y., Shahmorad, S., Talati, F.: Solving a class of two-dimensional linear and nonlinear Volterra integral equations by the differential transform method. J. Comput. Appl. Math. **228**, 70–76 (2009)
25. Tricomi, F.G.: Integral equations. Dover Publications, New York (1982)