Applied Soft Computing 12 (2012) 2817–2827

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Simulation and evaluation of fuzzy differential equations by fuzzy neural network

Maryam Mosleh [1], Mahmood Otadi *

Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firuozkooh, Iran.

ARTICLE INFO

Article history:
Received 27 May 2010
Received in revised form 14 February 2011
Accepted 18 March 2012
Available online 5 April 2012

Keywords:
Fuzzy neural networks
Fuzzy differentialequations
Feedforward neural network
Learning algorithm

ABSTRACT

In this paper, a novel hybrid method based on learning algorithm of fuzzy neural network for the solution of differential equation with fuzzy initial value is presented. Here neural network is considered as a part of large field called neural computing or soft computing. The model finds the approximated solution of fuzzy differential equation inside of its domain for the close enough neighborhood of the fuzzy initial point. We propose a learning algorithm from the cost function for adjusting of fuzzy weights. Finally, we illustrate our approach by numerical examples and an application example in engineering.

1. Introduction

Proper design for engineering applications requires detailed information of the system-property distributions such as temperature, velocity, and density in space and time domain [8–10]. This information can be obtained by either experimental measurement or computational simulation. Although experimental measurement is reliable, it needs a lot of labor efforts and time. Therefore, the computational simulation has become a more and more popular method as a design tool since it needs only a fast computer with a large memory. Frequently, those engineering design problems deal with a set of differential equations (DEs), which have to numerically solved such as heat transfer, solid and fluid mechanics. Numerical methods of predictor–corrector, Runge-Kutta, finite difference, finite element, finite volume, boundary element, spectral and collocation provide a strategy by which we can attack many problems in applied mathematics, where we simulate a real-word problem with a differential equation, subject to some initial or boundary conditions. In the finite difference and finite element methods we approximate the solution by using the numerical operators of the function's derivatives and finding the solution at specific preassigned grids [49]. The linearity is assumed for the purposes of evaluating the derivatives. Although such an approximation method is conceptually easy to understand, it has a number of shortcomings. Obviously, it is difficult to apply for systems with irregular geometry or unusual boundary conditions. Predictor–corrector and Runge-Kutta methods are widely applied over preassigned grid points to solve ordinary differential equations [31]. In the spectral and collocation approaches a truncated series of the specific orthogonal functions (basis functions) are used for finding the approximated solution of the DE. In the spectral and collocation techniques the role of trial functions as a basis function is important. The trial functions used in spectral methods are chosen from various classes of Jacobian polynomials [18], still the discretization meshes are preassigned. Neural network model is used to approximate the solutions of DEs for the entire domains. In 1990 the authors of [33] used parallel computers to solve a first order differential equation with Hopfield neural network models. Meade and Fernandez [38,39] solved linear and nonlinear ordinary differential equations using feed forward neural networks architecture and B_1-splines. Leephakpreeda [34] applied neural network model and linguistic model as universal approximators for any nonlinear continuous functions. With this outstanding capability, the solution of DEs can be approximated by the appropriate neural network model and linguistic model within an arbitrary accuracy.

When a physical problem is transformed into a deterministic initial value problem

$$\begin{cases} \dfrac{dy(x)}{dx} = f(x, y), \\ y(a) = A, \end{cases} \tag{1}$$

We usually cannot be sure that this modelling is perfect. The initial value may not be known exactly and the function f may contain unknown parameters. If the nature of errors is random, then instead of a deterministic problem (1) we get a random differential equation with random initial value and/or random coefficients. But if the underlying structure is not probabilistic, e.g., because of subjective

* Corresponding author. Tel.: +98 912 6964202.
E-mail addresses: mosleh@iaufb.ac.ir (M. Mosleh), otadi@iaufb.ac.ir (M. Otadi).
[1] Tel.: +98 912 6076308.

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2012.03.041

choice, then it may be appropriate to use fuzzy numbers instead of real random variables.

The topic of Fuzzy Differential Equations (FDEs) has been rapidly growing in recent years. The fuzzy initial value problem have been studied by several authors [1,2,6,7,44,40,11,14,51]. The concept of fuzzy derivative was first introduced by Chang and Zadeh [13], it was followed up by Dubois and Prade [15] who used the extension principle in their approach. Other methods have been discussed by Puri and Ralescu [43] and by Goetschel and Voxman [17]. Fuzzy differential equations were first formulated by Kaleva [28] and Seikkala [46] in time dependent form. Kaleva had formulated fuzzy differential equations, in terms of Hukuhara derivative [28]. Buckley and Feuring [12] have given a very general formulation of a fuzzy first-order initial value problem. They first find the crisp solution, make it fuzzy and then check if it satisfies the FDE. In [41,16], investigated the existence and uniqueness of solution for fuzzy random differential equations with non-lipschitz coefficients and fuzzy differential equations with piecewise constant argument.

In this work we propose a new solution method for the approximated solution of fuzzy differential equations using innovative mathematical tools and neural-like systems of computation. This hybrid method can result in improved numerical methods for solving fuzzy initial value problems. In this proposed method, fuzzy neural network model (FNNM) is applied as universal approximator. We use fuzzy trial function, this fuzzy trial function is a combination of two terms. A first term is responsible for the fuzzy initial while the second term contains the fuzzy neural network adjustable parameters to be calculated. The main aim of this paper is to illustrate how fuzzy connection weights are adjusted in the learning of fuzzy neural networks by the back-propagation-type learning algorithms [24,27] for the approximated solution of fuzzy differential equations. Our fuzzy neural network in this paper is a three-layer feedforward neural network where connection weights and biases are fuzzy numbers. The remaining part of the paper is organized as follows. In Section 2, we discuss some basic definitions. Also, we briefly review relevant definition of the architecture of fuzzy neural networks. Section 3 gives details of problem formulation and the way to construct the fuzzy trial function and training of fuzzy neural network for finding the unknown adjustable coefficients. Also, training of partially fuzzy neural network for finding the unknown adjustable coefficients and numerical examples are discussed in this section and conclusion is in final section.

2. Preliminaries

In this section the most basic notations used in fuzzy calculus are introduced. We start by defining the fuzzy number.

Definition 1. A fuzzy number is a fuzzy set $u : \mathbb{R}^1 \longrightarrow I = [0, 1]$ which satisfies

i. u is upper semi-continuous.
ii. $u(x) = 0$ outside some interval $[a, d]$.
iii. There are real numbers $b, c : a \le b \le c \le d$ for which

1. $u(x)$ is monotonic increasing on $[a, b]$,
2. $u(x)$ is monotonic decreasing on $[c, d]$,
3. $u(x) = 1, b, \le x \le c$.

The set of all the fuzzy numbers (as given by Definition 1) is denoted by E^1.

An alternative definition which yields the same E^1 is given by Kaleva [28].

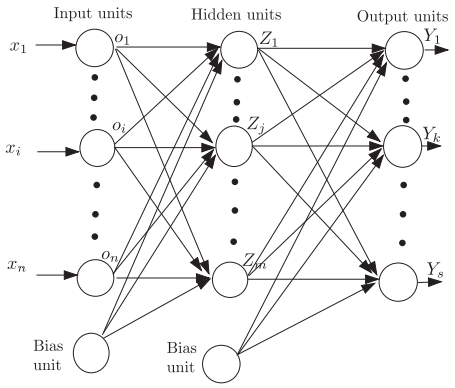

Fig. 1. Multiple layer feed-forward FNNM.

Definition 2. A fuzzy number u is a pair $(\underline{u}, \overline{u})$ of functions $\underline{u}(r), \overline{u}(r); 0 \le r \le 1$ which satisfy the following requirements:

i. $\underline{u}(r)$ is a bounded monotonic increasing left continuous function on $(0, 1]$ and right continuous at 0.
ii. $\overline{u}(r)$ is a bounded monotonic decreasing left continuous function on $(0, 1]$ and right continuous at 0.
iii. $\underline{u}(r) \le \overline{u}(r), 0 \le r \le 1$.

This fuzzy number space as shown in [50], can be embedded into the Banach space $B = \overline{C}[0, 1] \times \overline{C}[0, 1]$ where the metric is usually defined as

$$\|(u, v)\| = max\{\sup_{0\le r\le 1} |u(r)|, \sup_{0\le r\le 1} |v(r)|\},$$

for arbitrary $(u, v) \in \overline{C}[0, 1] \times \overline{C}[0, 1]$.

Artificial neural networks are an exciting form of artificial intelligence which mimic the learning process of the human brain in order to extract patterns from historical data [4,47]. For many years this technology has been successfully applied to a wide variety of real-word applications [42]. Simple perceptrons need a teacher to tell the network what the desired output should be. These are supervised networks. In an unsupervised net, the network adapts purely in response to its inputs. Such networks can learn to pick out structure in their input. Fig. 1 shows typical three-layered perceptron. Multi-layered perceptrons with more than three layers, use more hidden layers [21,29]. Multi-layered perceptrons correspond the input units to the output units by a specific nonlinear mapping [48]. From Kolmogorov existence theorem we know that a three-layered perceptron with $n(2n + 1)$ nodes can compute any continuous function of n variables [22,35]. The accuracy of the approximation depends on the number of neurons in the hidden layer and does not depend on the number of the hidden layers [32].

Before describing a fuzzy neural network architecture, we denote real numbers and fuzzy numbers by lowercase letters (e.g., $a, b, c, \ldots$) and uppercase letters (e.g., $A, B, C, \ldots$), respectively.

Our fuzzy neural network is a three-layer feedforward neural network where connection weights, biases and targets are given as fuzzy numbers and inputs are given as real numbers. For convenience in this discussion, FNNM with an input layer, a single hidden layer, and an output layer in Fig. 1 is represented as a basic structural architecture. Here, the dimension of FNNM is denoted by the number of neurons in each layer, that is $n \times m \times s$, where m, n and s are the number of neurons in the input layer, the hidden layer and the output layer, respectively. The architecture of the model shows how FNNM transforms the n inputs $(x_1, \ldots, x_i, \ldots, x_n)$ into the s outputs $(Y_1, \ldots, Y_k, \ldots, Y_s)$ throughout the m hidden neurons $(Z_1, \ldots, Z_j, \ldots, Z_m)$, where the cycles represent the neurons in each layer. Let B_j be the bias for neuron Z_j, C_k be the bias for neuron Y_k,