Design and Simulation of Intelligent Systems and Neural Networks

Edward Penell
Editor

Design and Simulation of Intelligent Systems and Neural Networks

Edited by Edward Penell

AURIS
Reference

www.aurisreference.com

Design and Simulation of Intelligent Systems and Neural Networks
Edited by Edward Penell

Published by Auris Reference ltd.
www.aurisreference.com

United Kingdom

Edition 2014

Design and Simulation of Intelligent Systems and Neural Networks
Edited by Edward Penell

ISBN: 978-1-78154-398-6

Printed in United Kingdom

Design and Simulation of Intelligent Systems and Neural Networks

Preface

In computer science and related fields, artificial neural networks are computational models inspired by animals' central nervous systems that are capable of machine learning and pattern recognition. For example, in a neural network for handwriting recognition, a set of input neurons may be activated by the pixels of an input image representing a letter or digit. The activations of these neurons are then passed on, weighted and transformed by some function determined by the network's designer, to other neurons, etc., until finally an output neuron is activated that determines which character was read.

Like other machine learning methods, neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition.

The first chapter investigates the solution of Ordinary Differential Equations (ODEs) with initial conditions using Regression Based Algorithm (RBA) and compares the results with arbitrary- and regression-based initial weights for different numbers of nodes in hidden layer. Here, we have used feed forward neural network and error back propagation method for minimizing the error function and for the modification of the parameters (weights and biases). Initial weights are taken as combination of random as well as by the proposed regression based model. The second chapter deals with treatment of slaughterhouse wastewater by conducting a laboratory scale sequencing batch reactor (SBR) with different input characterized samples, and the experimental results are explored for the formulation of feedforward backpropagation artificial neural network (ANN) to predict combined removal efficiency of chemical oxygen demand (COD) and ammonia nitrogen (-N). Recently, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing. We discussed this in the chapter third. A novel approach based on the neural network (NN) ensemble technique is formulated and used for development of a NN stochastic convection parameterization for climate and numerical weather prediction

(NWP) models. In the fourth chapter we discussed this. The fifth chapter intends to propose identification methodologies for multistorey shear buildings using the powerful technique of Artificial Neural Network (ANN) models which can handle fuzzified data. Identification with crisp data is known, and also neural network method has already been used by various researchers for this case. Here, the input and output data may be in fuzzified form. This is because in general we may not get the corresponding input and output values exactly (in crisp form), but we have only the uncertain information of the data. Multipath mitigation is a long-standing problem in global positioning system (GPS) research and is essential for improving the accuracy and precision of positioning solutions. In this work, we consider multipath error estimation as a regression problem and propose a unified framework for both code and carrier-phase multipath mitigation for ground fixed GPS stations. In the sixth chapter we discussed this. A hybrid learning scheme (ePSO-BP) to train Chebyshev Functional Link Neural Network (CFLNN) for classification is presented. The proposed method is referred as hybrid CFLNN (HCFLNN). The HCFLNN is a type of feed-forward neural networks which have the ability to transform the nonlinear input space into higher dimensional-space where linear separability is possible. Moreover, the proposed HCFLNN combines the best attribute of particle swarm optimization (PSO), back propagation learning (BP learning), and functional link neural networks (FLNNs). This we discussed in the seventh chapter. The eighth chapter introduces two unsupervised learning methods for analyzing functional magnetic resonance imaging (fMRI) data based on hidden Markov model (HMM). HMM approach is focused on capturing the first-order statistical evolution among the samples of a voxel time series, and it can provide a complimentary perspective of the BOLD signals. Two-state HMM is created for each voxel, and the model parameters are estimated from the voxel time series and the stimulus paradigm.

Editor

Contents

Chapter 1

COMPARISON OF ARTIFICIAL NEURAL NETWORK ARCHITECTURE IN SOLVING ORDINARY DIFFERENTIAL EQUATIONS

Susmita Mall and S. Chakraverty

Department of Mathematics, National Institute of Technology, Rourkela, Odisha-769008, India

ABSTRACT

This paper investigates the solution of Ordinary Differential Equations (ODEs) with initial conditions using Regression Based Algorithm (RBA) and compares the results with arbitrary- and regression-based initial weights for different numbers of nodes in hidden layer. Here, we have used feed forward neural network and error back propagation method for minimizing the error function and for the modification of the parameters (weights and biases). Initial weights are taken as combination of random as well as by the proposed regression based model. We present the method for solving a variety of problems and the results are compared. Here,

the number of nodes in hidden layer has been fixed according to the degree of polynomial in the regression fitting. For this, the input and output data are fitted first with various degree polynomials using regression analysis and the coefficients involved are taken as initial weights to start with the neural training. Fixing of the hidden nodes depends upon the degree of the polynomial. For the example problems, the analytical results have been compared with neural results with arbitrary and regression based weights with four, five, and six nodes in hidden layer and are found to be in good agreement.

INTRODUCTION

Differential equations play vital role in various fields of engineering and science. The exact solution of differential equations may not be always possible [1]. So various types of well known numerical methods such as Euler, Runge-kutta, Predictor-Corrector, finite element, and finite difference methods, are used for solving these equations. Although these numerical methods provide good approximations to the solution, but these may be challenging for higher dimension problems. In recent years, many researchers tried to find new methods for solving differential equations. As such here Artificial Neural Network (ANN) based models are used to solve ordinary differential equations with initial conditions.

Lee and Kang [2] first introduced a method to solve first order differential equation using Hopfield neural network models. Then, another approach by Meade and Fernandez [3, 4] has been proposed for both linear and nonlinear differential equations using -splines and feed forward neural network. Artificial neural networks based on Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization technique for solving ordinary and partial differential equations have been excellently presented by Lagaris et al. [5]. Also Lagaris et al. [6] investigated neural network methods for boundary value problems with irregular boundaries. Parisi et al. [7] presented unsupervised feed forward neural network for the solution of differential equations. The potential of the hybrid and optimization technique to deal with differential equation of lower order as well as higher order has been presented by Malek and Shekari Beidokhti [8]. Choi and Lee [9] discussed comparison of generalizing ability on solving differential equation using back propagation and reformulated radial basis function network. Yazdi et al. [10] used unsupervised kernel least mean square algorithm for solving ordinary differential equations. A new algorithm for solving matrix Riccati differential equations has been developed by Selvaraju and Abdul Samant [11]. He et al. [12] investigated a class of partial differential equations using multilayer neural network. Kumar and Yadav [13] surveyed multilayer perceptrons and radial basis function neural network methods for the solution of differential equations. Tsoulos et al. [14] solved differential equations with neural networks using a scheme based on grammatical evolution. Numerical solution of elliptic partial differential equation using radial basis function neural networks has been presented by Jianyu et al. [15]. Shirvany et al. [16] proposed multilayer perceptron and radial basis function (RBF) neural networks with a new unsupervised training method for numerical solution of partial differential equations. Mai-Duy and Tran-Cong [17] discussed numerical solution of differential equations using multiquadric radial basis function networks. Fuzzy linguistic model in neural network to solve differential equations is presented by Leephakpreeda [18]. Franke and Schaback [19] solved partial differential equations by collocation using radial basis functions. Smaoui and Al-Enezi [20] presented the dynamics of two nonlinear partial differential equations using artificial neural networks. Differential equations with genetic programming have been analyzed by Tsoulos and Lagaris [21]. McFall and Mahan [22] used artificial neural network for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. Hoda and Nagla [23] solved mixed boundary value problems using multilayer perceptron neural network method.

As per the review of the literatures, it reveals that authors have taken the parameters (weights/biases) as arbitrary (random) and the numbers of nodes in hidden layer are considered by trial and error method. In this paper, we propose a method for solving ordinary differential equations using feed forward neural network as a basic approximation element and error back propagation algorithm [24, 25] by fixing hidden nodes as per the required accuracy. The trial solution of the model is generated by training the algorithm. The approximate solution by ANN has many benefits compared with traditional numerical methods. The ANN trial solution is written as sum of two terms, first one satisfies initial/boundary conditions and the second part involves regression based neural network with adjustable parameters. The computational complexity does not increase considerably with the number of sampling points. The method is general so it can be applied to solve linear and nonlinear ordinary and partial differential equations. The modification of parameters has been done without direct use of optimization technique. For which computation of the gradient of error with respect to the network parameters is required. A regression based artificial neural network with combinations of initial weights (arbitrary and regression based) in the connections is first proposed by Chakraverty et al. [26] and then by Singh et al. [27]. Here, number of nodes in hidden layer may be fixed according to the degree of polynomial required for the accuracy. We have considered a first order and an application problem such as damped free vibration problem to show the comparison of different ANN models. Mall and Chakraverty [28] proposed regression-based neural network model for solving ordinary differential equations.

Rest of the paper is organized as follows. In Section 2, we describe the general formulation of the proposed approach and computation of gradient of the error function. Section 3 gives details of problem formulation and construction of the appropriate form of trial solution. The proposed regression based artificial neural network method has been presented in Section 4. Numerical examples and its results are presented in Section 5. In this section, we compare arbitrary and regression based weight results and those are shown graphically. Section 6 incorporates the discussion and analysis part. Lastly conclusion is outlined in Section 7.

GENERAL FORMULATION FOR DIFFERENTIAL EQUATIONS

Let us consider the following general differential equations which represent both ordinary and partial differential equations [4]:

$$G\left(x, \psi(x), \nabla\psi(x), \nabla^2\psi(x) \cdots\right) = 0, \quad x \in D, \tag{1}$$

Subject to some initial or boundary conditions, where ,

$x = (x_1, x_2, \ldots, x_n) \in R^n,\ D \subset R^n$ denotes the domain, and $\psi(x)$ is the solution to be computed. Here, *G* is the function which defines the structure of the differential equation and ∇ is a differential operator. For the solution of the differential equation, a discretized domain $\overline{D}$ over finite set of points in *D* is considered. Thus, the problem transformed into the system of equations as follows:

$$G\left(x_i, \psi(x_i), \nabla\psi(x_i), \nabla^2\psi(x_i)\cdots\right) = 0, \quad x \in \overline{D}. \tag{2}$$

Let $\psi_t(x, p)$ denote the trail solution with adjustable parameters (weights, biases) , and then the problem may be formulated as

$$G(x_i, \psi_t(x_i, p), \nabla\psi_t(x_i, p), \ldots, \nabla^m\psi_t(x_i, p)\cdots) = 0. \tag{3}$$

Corresponding error function with respect to every input data is written as

$$\min_p \sum_{x_i \in \overline{D}} (G(x_i, \psi_t(x_i, p), \nabla\psi_t(x_i, p), \ldots, \nabla^m\psi_t(x_i, p)))^2. \tag{4}$$

Now, $\psi_t(x, p)$ may be written as the sum of two terms

$$\psi_t(x, p) = A(x) + F(x, N(x, p)), \tag{5}$$

where *A*(*x*) satisfies initial or boundary condition and contains no adjustable parameters, whereas *N*(*x, p*) is the output of feed forward neural network with the parameters *p* and input data *x* The second term *F*(*x, N*(*x, p*)) makes no contribution to initial or boundary but this is used to a neural network model whose weights and biases are adjusted to minimize the error function.

Computation of the Gradient

The error computation not only involves the outputs but also the derivatives of the network output with respect to its inputs. So, it requires finding out the gradient of the network derivatives with respect to its inputs. Let us now consider a multilayered perceptron with one input node, a hidden layer with nodes (fixed number of nods as proposed), and one output unit. For the given inputs $x = (x_1, x_2, \ldots x_n)$, the output is given by

$$N(x, p) = \sum_{j-1}^{m} v_j \sigma(z_j), \tag{6}$$

where $z_j = \sum_{i-1}^{n} w_{ji} x_i + u_j, w_{ji}$ denotes the weight from input unit i to the hidden unit j, v_j denotes weight from the hidden unit j to the output unit, u_j denotes the biases, and $\sigma(z_j)$ is the sigmoid activation function.

The derivatives of $N(x, p)$with respect to input x_i is

$$\frac{\partial^k N}{\partial x_i^k} = \sum_{j=1}^{m} v_j w_{ji}^k \sigma_j^{(k)}, \tag{7}$$

where $\sigma = \sigma(z_j)$ and $\sigma^{(k)}$ denotes the kth order derivative of sigmoid function.

Let N_θ denote the derivative of the network with respect to its inputs and then we have the following relation [4]:

$$N_\vartheta = D^n N = \sum_{i=1}^{n} v_i P_i \sigma_i^{(n)}, \tag{8}$$

Where

$$P_j = \prod_{k=1}^{n} w_{jk}^{\lambda_k}, \quad \kappa = \sum_{i=1}^{n} \lambda_i. \tag{9}$$

The derivative of N_ϑ with respect to other parameters may be obtained as

$$\frac{\partial N_\vartheta}{\partial v_j} = P_j \sigma_j^{(\kappa)}, \tag{10}$$

$$\frac{\partial N_\vartheta}{\partial u_j} = v_j P_j \sigma_j^{(\kappa+1)}, \tag{11}$$

$$\frac{\partial N_\vartheta}{\partial w_{ji}} = x_i v_j P_j \sigma_j^{(\kappa+1)} + v_j \lambda_i w_{ji}^{\lambda_i - 1} \left(\prod_{k=1, k \neq i} w_{ji}^{\lambda_k} \right) \sigma_j^{(\kappa)}. \tag{12}$$

FORMULATION OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATION

Let us consider first order ordinary differential equation as below

$$\frac{d\psi}{dx} = f(x, \psi), \quad x \in [a, b], \tag{13}$$

with initial condition $\psi(a) = A.$.

In this case, the ANN trail solution may be written as

$$\psi_t(x,p) = A + (x-a)N(x,p), \tag{14}$$

where $N(x, p)$ is the neural output of the feed forward network with one input data x with parameters p. The trial solution $\psi_t(x,p)$ satisfies the initial condition. We differentiate the trial solution $\psi_t(x,p)$ to get

$$\frac{d\psi_t(x,p)}{dx} = N(x,p) + (x-a)\frac{dN(x,p)}{dx}. \tag{15}$$

For evaluating the derivative term in the right hand side of (15), we use (5)–(11).

The error function for this case may be formulated as

$$E(p) = \sum_{i-1}^{n}\left(\frac{d\psi_t(x_i,p)}{dx} - f(x_i,\psi_t(x_i,p))\right)^2. \tag{16}$$

The weights from input to hidden are modified according to the following rule

$$w_{ji}^{r+1} = w_{ji}^{r} - \eta\left(\frac{\partial E}{\partial w_{ji}^{r}}\right), \tag{17}$$

Where

$$\frac{\partial E}{\partial w_{ji}^{r}} = \frac{\partial}{\partial w_{ji}^{r}}\left(\sum_{i-1}^{n}\left(\frac{d\psi_t(x_i,p)}{dx} - f(x_i,\psi_t(x_i,p))\right)^2\right). \tag{18}$$

Here, η is the learning rate and r is the iteration step. The weights from hidden to output layer may be updated in a similar formulation as done for input to hidden.