Expert Systems with Applications 42 (2015) 855–863

Contents lists available at ScienceDirect

Expert Systems with Applications

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

# Back propagation neural network with adaptive differential evolution algorithm for time series forecasting

Lin Wang [a], Yi Zeng [a], Tao Chen [b,*]

[a] School of Management, Huazhong University of Science and Technology, Wuhan 430074, China
[b] College of Public Administration, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

*Article history:*
Available online 27 August 2014

*Keywords:*
Time series forecasting
Back propagation neural network
Differential evolution algorithm

ABSTRACT

The back propagation neural network (BPNN) can easily fall into the local minimum point in time series forecasting. A hybrid approach that combines the adaptive differential evolution (ADE) algorithm with BPNN, called ADE–BPNN, is designed to improve the forecasting accuracy of BPNN. ADE is first applied to search for the global initial connection weights and thresholds of BPNN. Then, BPNN is employed to thoroughly search for the optimal weights and thresholds. Two comparative real-life series data sets are used to verify the feasibility and effectiveness of the hybrid method. The proposed ADE–BPNN can effectively improve forecasting accuracy relative to basic BPNN, autoregressive integrated moving average model (ARIMA), and other hybrid models.



## 1. Introduction

Time series forecasting is an important area in forecasting. One of the most widely employed time series analysis models is the autoregressive integrated moving average (ARIMA), which has been used as a forecasting technique in several fields, including traffic (Kumar & Jain, 1999), energy (Ediger & Akar, 2007), economy (Khashei, Rafiei, & Bijari, 2013), tourism (Chu, 2008), and health (Yu, Kim, & Kim, 2013). ARIMA has to assume that a given time series is linear (Box & Jenkins, 1976). However, time series data in real-world settings commonly have nonlinear features under a new economic era (Lee & Tong, 2012; Liu & Wang, 2014a, 2014b; Matias & Reboredo, 2012). Consequently, ARIMA may be unsuitable for most nonlinear real-world problems (Khashei, Bijari, & Ardali, 2009; Zhang, Patuwo, & Hu, 1998). Artificial neural networks (ANNs) have been extensively studied and used in time series forecasting (Adebiyi, Adewumi, & Ayo, 2014; Bennett, Stewart, & Beal, 2013; Geem & Roper, 2009; Zhang, Patuwo, & Hu, 2001; Zhang & Qi, 2005). Zhang et al. (1998) presented a review of ANNs. The advantages of ANNs are their flexible nonlinear modeling capability, strong adaptability, as well as their learning and massive parallel computing abilities (Ticknor, 2013). Specifying a particular model form is unnecessary for ANNs; the model is instead adaptively formed based on the features presented by the data. This data-driven approach is suitable for many empirical data sets, wherein theoretical guidance is unavailable to suggest an appropriate data generation process. The forward neural network is the most widely used ANNs. Meanwhile, the back propagation neural network (BPNN) is one of the most utilized forward neural networks (Wang, Zeng, Zhang, Huang, & Bao, 2006). BPNN, also known as error back propagation network, is a multilayer mapping network that minimizes an error backward while information is transmitted forward. A single hidden layer BPNN can generally approximate any nonlinear function with arbitrary precision (Aslanargun, Mammadov, Yazici, & Yolacan, 2007). This feature makes BPNN popular for predicting complex nonlinear systems.

BPNN is well known for its back propagation-learning algorithm, which is a mentor-learning algorithm of gradient descent, or its alteration (Zhang et al., 1998). According to the theory, the connection weights and thresholds of a network are randomly initialized first. Then, by using the training sample, the connection weights and thresholds of the network are adjusted to minimize the mean square error (MSE) of the network output value and actual value through gradient descent. When the MSE achieves the goal setting, the connection weights and thresholds are determined, and the training process of the network is finished. However, one flaw of this learning algorithm is that the final training result depends on the initial connection weights and thresholds to a large extent. Hence, the training result easily falls into the local minimum point rather than into the global optimum; thus, the network cannot forecast precisely. To overcome this shortcoming, many researchers have proposed different methods to optimize

* Corresponding author.
*E-mail addresses:* wanglin982@gmail.com (L. Wang), zengy200810@126.com (Y. Zeng), chentao15@163.com (T. Chen).

http://dx.doi.org/10.1016/j.eswa.2014.08.018
0957-4174/

the initial connection weights and thresholds of traditional BPNN. Yam and Chow (2000) proposed a linear algebraic method to select the initial connection weights and thresholds of BPNN. Intelligent evolution algorithms, such as the genetic algorithm (GA) (Irani & Nasimi, 2011) and particle swarm optimization (PSO) (Zhang, Zhang, Lok, & Lyu, 2007), have also been used to select the initial connection weights and thresholds of BPNN. The proposed models are superior to traditional BPNN models in terms of convergence speed or prediction accuracy.

As a novel evolutionary computational technique, the differential evolution algorithm (DE) performs better than other popular intelligent algorithms, such as GA and PSO, based on 34 widely used benchmark functions (Vesterstrom & Thomsen, 2004). Compared with popular intelligent algorithms, DE has less complex genetic operations because of its simple mutation operation and one-on-one competition survival strategy. DE can also use individual local information and population global information to search for the optimal solution (Wang, Fu, & Zeng, 2012; Wang, Qu, Chen, & Yan, 2013; Zeng, Wang, Xu, & Fu, 2014). DEs and improved DEs are among the best evolutionary algorithms in a variety of fields because of their easy implementation, quick convergence, and robustness (Onwubolu & Davendra, 2006; Qu, Wang, & Zeng, 2013; Wang, He, & Zeng, 2012). However, only a few researchers have used the DE to select suitable BPNN initial connection weights and thresholds in time series forecasting. Therefore, this study uses adaptive DE (ADE) to select appropriate initial connection weights and thresholds for BPNN to improve its forecasting accuracy. Two real-life time series data sets with nonlinear and cyclic changing tendency features are employed to compare the forecasting performance of the proposed model with those of other forecasting models.

The remainder of this paper is organized as follows. Section 2 discusses the ADE–BPNN model, including theory of BPNN in time series forecasting and the ADE process. Section 3 presents two numerical examples. Section 4 concludes the study.

## 2. BPNN with DE

### 2.1. BPNN for time series forecasting

A single hidden layer BPNN consists of an input layer, a hidden layer, and an output layer as shown in Fig. 1. Adjacent layers are connected by weights, which are always distributed between −1 and 1. A systematic theory to determine the number of input nodes and hidden layer nodes is unavailable, although some heuristic approaches have been proposed by a number of researchers (Zhang & Subbarayan, 2002; Zhang et al., 1998). None of the choices, however, works efficiently for all problems. The most common means to determine the appropriate number of input and hidden nodes is via experiments or by trial and error based on the minimum mean square error of the test data (Hosseini, Luo, & Reynolds, 2006).

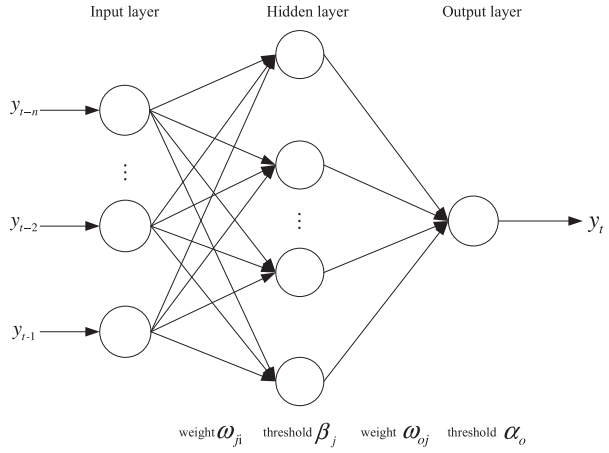


**Fig. 1.** Single hidden layer BPNN structure.

In the current study, a single hidden layer BPNN is used for one-step-ahead forecasting. Several past observations are used to forecast the present value. That is, the input is $y_{t-n}, y_{t-n+1}, \ldots, y_{t-2}, y_{t-1}$; and $y_t$ is the target output. The input and output values of the hidden layer are represented as Eqs. (1) and (2), respectively. The input and output values of the output layer are represented as Eqs. (3) and (4), respectively. The equations are given as follows:

$$I_j = \sum_{i=t-n}^{t-1} w_{ji} \times y_i + \beta_j \quad (j = 1, \ldots, h), \tag{1}$$

$$y_j = f_h(I_j) \quad (j = 1, \ldots, h), \tag{2}$$

$$I_o = \sum_{j=1}^{h} w_{oj} \times y_j + \alpha_o \quad (o = 1), \tag{3}$$

$$y_t = f_o(I_o) \quad (o = 1), \tag{4}$$

where $I$ denotes the input; $y$ denotes the output; $y_t$ is the forecasted value of point $t$; $n$ and $h$ denote the number of input layer nodes and hidden layer nodes, respectively; $w_{ji}$ denotes the connection weights of the input and hidden layers; and $w_{oj}$ denotes the connection weights of the hidden and output layers. $\beta_j$ and $\alpha_o$ are the threshold values of the hidden and output layers, respectively, which are always distributed between −1 and 1. $f_h$ and $f_o$ are the activation functions of the hidden and output layers, respectively. Generally, the activation function of each node in the same layer is the same. The most widely used activation function for the output layer is the linear function because the nonlinear activation function may introduce distortion to the predicted output. The logistic and hyperbolic functions are frequently used as the hidden layer activation functions (Zhang et al., 1998).

### 2.2. DE and ADE

#### 2.2.1. Standard DE

The standard DE consists of four main operations: initialization, mutation, crossover, and selection. Details are discussed as follows:

(1) Initialization: Real number coding is used for the DE. In this operation, several parameters, including population size $N$, length of chromosome $D$, scaling or mutation factor $F$, crossover rate $CR$, and the range of gene value $[U_{\min}, U_{\max}]$, are initialized. The population is randomly initialized with Eq. (5), as follows:

$$x_{ij} = U_{\min} + rand \times (U_{\max} - U_{\min}), \tag{5}$$

where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$, and $rand$ is a random number with a uniform probability distribution.

(2) Mutation: For each objective individual $x_i^G$, $i = 1, 2, \ldots, N$, the standard DE algorithm generates a corresponding mutated individual, which is expressed by Eq. (6):

$$v_i^{G+1} = x_{r_1}^G + F \times \left(x_{r_2}^G - x_{r_3}^G\right), \tag{6}$$

where the individual serial numbers $r_1$, $r_2$, and $r_3$ are different and randomly generated. None of the numbers is identical to the objective individual serial number $i$. Therefore, the population size $N \geqslant 4$. The scaling factor $F$, which controls the mutation degree, is within the range of [0,2], as mentioned