Neurocomputing 148 (2015) 136–142

Contents lists available at ScienceDirect

Neurocomputing

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

Evolving RBF neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm

Jiansheng Wu [a,*], Jin Long [b], Mingzhe Liu [c]

[a] School of Information Engineering, Wuhan University of Technology, Wuhan, 430070 Hubei, PR China
[b] Guangxi Research Institute of Meteorological Disasters Mitigation, Naning, Guangxi 535002, PR China
[c] State Key Laboratory of Geohazard Prevention and Geoenvironment Protection, Chengdu University of Technology, Chengdu 610059, PR China

ARTICLE INFO

Article history:
Received 6 April 2012
Received in revised form
4 October 2012
Accepted 8 October 2012
Available online 1 August 2014

Keywords:
Particle swarm optimization
Genetic algorithm
Radial basis function neural network
Rainfall prediction

ABSTRACT

In this paper, an effective hybrid optimization strategy by incorporating the adaptive optimization of particle swarm optimization (PSO) into genetic algorithm (GA), namely HPSOGA, is used for determining the parameters of radial basis function neural networks (number of neurons, their respective centers and radii) automatically. While this task depends upon operator's experience with trial and error due to lack of prior knowledge, or based on gradient algorithms which are highly dependent on initial values. In this paper, hybrid evolutionary algorithms are used to automatically build a radial basis function neural networks (RBF-NN) that solves a specified problem, related to rainfall forecasting in this case. In HPSOGA, individuals in a new generation are created through three approaches to improve the global optimization performance, which are elitist strategy, PSO strategy and GA strategy. The upper-half of the best-performing individuals in a population are regarded as elites, whereas the half of the worst-performing individuals are regarded as a swarm. The group constituted by the elites are enhanced by selection, crossover and mutation operation on these enhanced elites. HPSOGA is applied to RBF-NN design for rainfall prediction. The performance of HPSOGA is compared to pure GA in these basis function neural networks design problems, showing that the hybrid strategy is of more effective global exploration ability and to avoid premature convergence. Our findings reveal that the hybrid optimization strategy proposed here may be used as a promising alternative forecasting tool for higher forecasting accuracy and better generalization ability.

1. Introduction

Accurate and timely rainfall prediction is essential for the planning and management of water resources, in particular for flood warning systems because it can provide an effective information to help prevent casualties and damages caused by natural disasters. Rainfall prediction is very difficult to model since the complexity of the atmospheric processes involves a rather complex nonlinear pattern, for example pressure, temperature, wind speed and its direction, meteorological characteristics of the precipitation area and so on [24,26,28,30]. Neural network (NN) techniques have been recognized as more useful than conventional statistical forecasting models because they can map any non-linear function without understanding the physical laws and any assumptions of traditional statistical approaches required [11,18].

Many types of NN have been proposed for rainfall forecasting, among which two widely used categories are back propagation neural network (BPNN) [10] and radial basis function neural network (RBF-NN) [27]. RBF-NN can approximate any continuous function on a compact set to a desired degree of precision. However, the technique of finding the suitable paraments of radial functions is very complex because inappropriate network is inefficient and sensitive to over-fitting. Such a model might be doing well in predicting past incidents, but unable to predict future events.

In the last two decades many evolutionary algorithms have been proposed, such as genetic algorithm (GA) [7] and particle swarm optimization (PSO) [13], which has provided a more robust and efficient approach for solving complex real-world problems [30,31]. In contrast to traditional optimization methods, which emphasize accurate and exact computation for the global optimum of a continuous function and avoid being trapped into one of the local optima, but may fall down on achieving the global optimum [1,14].

GA is a heuristic optimization search algorithm simulating the evolutionary ideas of natural selection [7]. Since this type of

* Corresponding author.
E-mail addresses: wjsh2002168@163.com (J. Wu), long01@163.com (J. Long), liumz@cdut.edu.cn (M. Liu).

http://dx.doi.org/10.1016/j.neucom.2012.10.043
0925-2312/© 2014 Elsevier B.V. All rights reserved.

algorithm simultaneously evaluates many points in the search space, it is more likely to find the global solution of a given problem. In addition, it uses only a simple scalar performance measure that does not require or use derivative information, so methods classified as GA are easy to use and implement [5]. Similar to GA, PSO is also a heuristic algorithm based on population of random solutions. Its development was based on observations of the social behavior of animals such as bird flocking, fish schooling, and swarm theory. Compared to GA, the PSO has much more profound intelligent background and could be performed more easily [19,20]. It has memory, so knowledge of good solutions is retained by all particles; whereas in GA, previous knowledge of the problem is destroyed once the population changes. It has constructive cooperation between particles, particles in the swarm share information between them [17,29]. GA and PSO have been successfully applied to optimize neural network [12,24,28,29].

Recently, hybridization of evolutionary computation with local search has been investigated in many studies in order to overcome the local optimum problem of GA and PSO, to find a better optimal solution [12,15,25]. Juang developed hybrid GAPSO for recurrent network design. In his hybrid algorithm, individuals were created, not only by crossover and mutation operation in GA, but also by PSO [12]. Li et al. presented a gene selection method using combined PSO with GA using SVM [15]. Shi et al. presented a variable population size GA by the "dying probability" for the PSO individuals, synthesized the GA and PSO [25].

Different from the previous work, one of the main purposes in this paper is to develop an effective hybrid optimization strategy by incorporating PSO into GA for the best components of the RBF-NN, namely RBF-HPSOGA, to approximate a function representing a rainfall time-series. Results are compared with pure GA that also use the evolutionary approach to tune the RBF-NN.

The rest of this paper is organized as follows. Section 2 describes the RBF-NN. Section 3 describes the hybrid evolutionary algorithm for RBF-NN design. In Section 4, designs of RBF-NN by HPSOGA are simulated and applied to rainfall forecasting. Finally, conclusions are presented in Section 5.

2. Radial basis function neural network

Radial basis function (RBF) networks were introduced into the neural network literature by Broomhead and Lowe [2]. On the contrary to classical models (multilayer perceptrons, etc.), it is a network with local units which was motivated by the presence of many local response units in human brain [3,22]. Neurons with a locally tuned response characteristic can be found in several parts of the nervous system, for example cells in the auditory system selective to small bands of frequencies or cells in the visual cortex sensitive to bars oriented in a certain direction or other visual features within a small region of the visual field.

The basic architecture of a three-layered RBF-NN is shown in Fig. 1. The network is generally composed of three layers: an input layer, a single hidden layer of nonlinear processing neurons, and an output layer. The output of the RBF-NN is calculated according to

$$y_t = \sum_{i=1}^{N} \omega_{ti}\phi_i(x, c_i) = \sum_{i=1}^{N} \omega_{ti}\phi_i(\|x-c_i\|_2), \quad t = 1, 2, \ldots, m \tag{1}$$

where $x \in \Re^{n\times 1}$ is an input vector, $\phi_k(\cdot)$ is a radial basis function from $\Re^{n\times 1}$ (set of all positive real numbers) to $\Re$, $\|\cdot\|_2$ denotes the Euclidean norm, w_{ti} are the weights of the links that connects hidden neuron number i and output neuron number t in the output layer, N is the number of neurons in the hidden layer and $c_i \in \Re^{n\times 1}$ are the RBF centers in the input vector space [9].

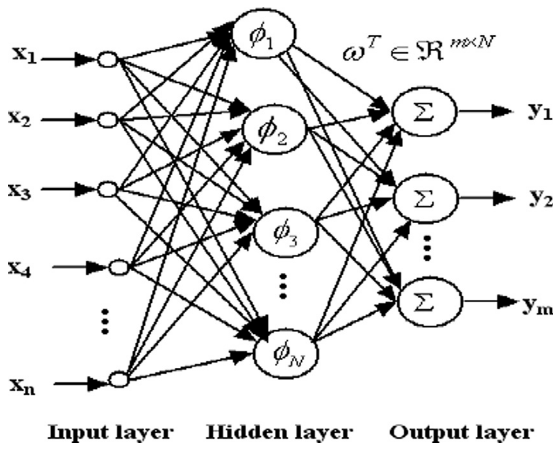

Fig. 1. The RBF-NN architecture.

The input layer is composed of input data. The hidden layer transforms the data from the input space to the hidden space using a non-linear function. The output layer, which is linear, yields the response of the network. A normalized Gaussian function usually is used as the radial basis function as follows:

$$\phi_i(x, c_i) = \exp\left(-\frac{\|x-c_i\|_2}{r_i^2}\right) \tag{2}$$

where r_i denote the radius of the i-th node. It has been proved that when enough units are provided, a RBF-NN can approximate any multivariate continuous function as much as desired [16]. Thus, the main problem in RBF NNs design concerns establishing the number of hidden neurons to use and their centers and radii. Generally, the RBF-NN can been acquired by two stages:

1. Determine the paraments of radial basis function, i.e., Gaussian center and radius, in which k-means clustering method was commonly used.
2. Determine the output weight ω by supervised learning method, in which least mean square or recursive least square was usually used.

The first stage is very crucial, because the number of hidden and location of center will affect the performance of RBF-NN. When this number is not sufficient, the approximation offered by the net is sensitive to over-fitting or poor performances. The values for the center, the radii and the weights are also important parameters. Very few parameters will lead to over-fitting, and while too much parameters can give even worst results. In RBF-NN application, establishing the number of neurons (that is, the number of centers and values related to them) is one of the most important tasks researchers have faced in this field. Thus, in order to obtain a good performance, network parameters need to be considered in coordination [6,21,23]. In this paper a hybird PSOGA algorithm is used to find the best components of the RBF-NN that approximate a function representing a rainfall timeseries.

3. Hybrid of PSO and GA for RBF-NN design

3.1. Encode

In this paper HPSOGA algorithm is used to establish the best components of the RBF-NN that approximate a function, including the number of hidden neurons, the centers, c_i, radii, r_i, and weights, ω_i. The hidden nodes are encoded as binary code string, 1 with connection to input and output nodes and 0 with no connection. The centers, radii