Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 69 (2006) 2268–2283
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Evolutionary system for automatically constructing and adapting radial basis function networks

Daniel Manrique*, Juan Ríos, Alfonso Rodríguez-Patón

Department Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain

Received 31 October 2004; received in revised form 2 June 2005; accepted 15 June 2005
Available online 20 December 2005
Communicated by H. Degaris

Abstract

This article presents a new system for automatically constructing and training radial basis function networks based on original evolutionary computing methods. This system, called Genetic Algorithm Radial Basis Function Networks (GARBFN), is based on two cooperating genetic algorithms. The first algorithm uses a new binary coding, called basic architecture coding, to get the neural architecture that best solves the problem. The second, which uses real coding, takes its inspiration from mathematical morphology theory and trains the architectures output by the binary genetic algorithm. This system has been applied to a laboratory problem and to breast cancer diagnosis. The results of these evaluations show that the overall performance of GARBFN is better than other related approaches, whether or not they are based on evolutionary techniques.

Keywords: Genetic algorithms; Radial basis function networks; Self-adaptive intelligent system; Breast cancer diagnosis

1. Introduction

One of the main fields of artificial intelligence research is the development of self-adaptive systems capable of transforming to solve different problem types [17]. On account of their pattern-based learning, data generalization and noise filtering capabilities [18,23], neural networks are commonly used within artificial intelligence to perform real-world tasks.

Radial basis function networks (RBFN) are a type of network that is very useful for pattern classification problems [27]. This is because, unlike the multilayer perceptron (MLP) [23] whose output is generated by the action of all the neurons in the network weighted by the weights of its connections, the output of a RBFN is mainly influenced by the hidden layer neuron, whose centre is closer to the input pattern. Therefore, RBFNs are local approximators, whereas MLPs are global approximators [13].

The benefits of using a RBFN are: (i) local data approximation uses few hidden units for any input; (ii) the hidden and output layer parameters can be trained separately using a hybrid algorithm, and (iii) only one, non-linear, hidden layer is used, whereas the output is linear. As only one hidden layer is used, they converge faster than a MLP with several hidden layers [27].

Despite the advantages of RBFNs, the design of an optimal architecture to solve a particular problem is far from being a straightforward matter [15]. Additionally, RBFNs trained according conventional gradient descent methods have been shown to be more likely to get trapped in local optima, and they are, therefore, less accurate than when applied to MLPs [4]. This is because, as far as RBFNs are concerned, apart from finding a set of weights for connections that best solve the problem, the centres of the radial basis functions of the hidden layer neurons have to be searched.

Several research papers have, therefore, focused on designing new approaches based on different search and optimization techniques to choose the best neural architecture to solve a given problem and to speed up the training process. Some of these studies deal with the

*Corresponding author. Tel.: +34 91 336 6907; fax: +34 352 4819.
E-mail addresses: dmanrique@fi.upm.es (D. Manrique), jrios@fi.upm.es (J. Ríos), arpaton@fi.upm.es (A. Rodríguez-Patón).

0925-2312/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.06.018

so-called incremental algorithms [26], which start from a predefined neural architecture and then dynamically add and remove neural connections during the training process. The performance of these algorithms is low and they tend to converge prematurely depending on the original architecture chosen. They cannot, therefore, guarantee a good solution [22].

To surmount the problem of trapping in local optima and improve the accuracy of the results of applying gradient backpropagation to RBFN, research has been conducted aimed at building hybrid training algorithms [7]. These algorithms combine an unsupervised search of the hidden layer neuron centres using clustering algorithms, such as *k*-means, or improved versions, like moving *k*-means, with gradient backpropagation to get the weights of the connections [19]. However, these approaches are still beset by the very same weakness of local optima trapping, because they use derivatives-based optimization methods.

Other more promising studies originate from the identification of synergies between evolutionary algorithms and artificial neural networks that can be combined in various ways. These include works related to the genetic adaptation of the internal structure of the network [3,16,25]. Genetic algorithms have also been used to partially replace the network learning method, first applying genetic algorithms to accomplish global search until a point near the solution is reached and then running a local search with classical gradient descent methods to get the optimum solution [5,27]. Other numerical approaches, like regularized orthogonal least squares can also be used [6] instead of gradient descent methods. The snag with all these approaches is that it is not known what is the best time to switch from global to local search.

For any chosen evolutionary optimization approach, the way in which the neural networks that make up the search space are encoded is a crucial step in automatic network design [14]. Therefore, several approaches have been developed to produce efficient codifications of artificial neural networks, and specifically RBFNs. The first of these is the direct binary encoding of network configuration [10], where each bit determines the presence or absence of a single connection. There are two major problems with this approach. First, convergence performance is degraded as the number of neurons in the hidden layer increases because the search space is much larger. Second, direct encoding methods cannot prevent illegal points in the search space (architectures that are not permitted for a RBFN). At the other end of the scale from the direct encoding methods are other approaches to RBFN codification, where there is no direct correspondence between each bit of the string and each connection of the neural architecture. These are called indirect encoding methods, of which the graph generation system is the method for which the best results have been reported [16]. This approach is based on the binary codification of grammars that describe the architecture of the network and prevent the codification of illegal neural architectures. The problem here is that a one-bit variation in a string results in a totally different network. This degrades the convergence process of the genetic algorithm that uses this codification.

Training RBFNs can be seen as an optimization problem, where the mean square error has to be minimized by adjusting the values of the weights of the connections and the centres of the neurons in the hidden layer. Evolutionary algorithms are therefore a suitable option for dealing with this problem. Michalewicz states that if a problem is real-valued in nature, then a real number genetic algorithm is faster and more precise than a binary encoded genetic algorithm [20]. Therefore, genetic algorithms using real number codification to represent the weights of the neural network can be expected to yield the best results. There is a variety of techniques for handling real-coded genetic algorithms. Radcliffe's flat crossover chooses parameters for an offspring by uniformly picking parameter values from (inclusively) the two parents' parameter values [24]. BLX-α was then proposed [11] to work around the premature convergence problems of this operator. BLX-α uniformly picks values that lie between two points that contain the two parents and may extend equally on either side of the interval defined by the parents. This new method, however, is very slow at approximating to the optimum because the extension of the interval defined by the parents is determined by a static, user-specified parameter α set at the start of the run. Another important crossover technique for real-coded genetic algorithms is UNDX [21], which can optimize functions by generating offspring using the normal distribution defined by three parents. The problem here is the high computational cost required to calculate the normal distribution. Other research combines statistical methods, pruning and real-coded genetic algorithms [13], although the problem is, again, the computational cost of calculating the Bayesian regularization on which this algorithm is based.

This paper presents a new evolutionary system, called Genetic Algorithm Radial Basis Function Networks (GARBFN), which automatically designs and trains RBFNs to solve a given problem stated as a set of training patterns. The work presented here is the basis for automatically building RBFN-based self-adaptive intelligent systems. GARBFN consists of a binary-coded genetic algorithm that searches for neural architectures in combination with a hybrid training method that employs a *k*-mean clustering algorithm to ascertain the centres of the neurons in the hidden layer, and a real-coded genetic algorithm rather than any of the other conventional methods to adjust the weights of the connections.

The binary-coded genetic algorithm employs the basic architectures codification method [3], which has been adapted to work on radial basis function architectures. A specialized binary crossover operator—the RBFN crossover (RBFN-X)—has also been designed to work with the proposed codification method. This operator outperforms