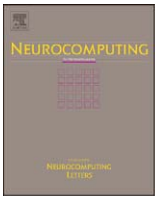
Contents lists available at ScienceDirect

Neurocomputing

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



Parallel multiobjective memetic RBFNNs design and feature selection for function approximation problems

A. Guillén [a,*], H. Pomares [b], J. González [b], I. Rojas [b], O. Valenzuela [b], B. Prieto [b]

[a] *Department of Informatics, University of Jaen, Spain*
[b] *Department of Computer Technology and Architecture, University of Granada, Spain*

ARTICLE INFO

Available online 23 June 2009

Keywords:
Parallel genetic algorithms
RBF
Neural networks
Function approximation
RBFNN
MPI

ABSTRACT

The design of radial basis function neural networks (RBFNNs) still remains as a difficult task when they are applied to classification or to regression problems. The difficulty arises when the parameters that define an RBFNN have to be set, these are: the number of RBFs, the position of their centers and the length of their radii. Another issue that has to be faced when applying these models to real world applications is to select the variables that the RBFNN will use as inputs. The literature presents several methodologies to perform these two tasks separately, however, due to the intrinsic parallelism of the genetic algorithms, a parallel implementation will allow the algorithm proposed in this paper to evolve solutions for both problems at the same time. The parallelization of the algorithm not only consists in the evolution of the two problems but in the specialization of the crossover and mutation operators in order to evolve the different elements to be optimized when designing RBFNNs. The subjacent genetic algorithm is the non-sorting dominated genetic algorithm II (NSGA-II) that helps to keep a balance between the size of the network and its approximation accuracy in order to avoid overfitted networks. Another of the novelties of the proposed algorithm is the incorporation of local search algorithms in three stages of the algorithm: initialization of the population, evolution of the individuals and final optimization of the Pareto front. The initialization of the individuals is performed hybridizing clustering techniques with the mutual information (MI) theory to select the input variables. As the experiments will show, the synergy of the different paradigms and techniques combined by the presented algorithm allow to obtain very accurate models using the most significant input variables.

1. Introduction

The problem of function approximation, also known as non-linear regression, has been successfully tackled using radial basis function neural networks (RBFNNs) [32]. Formally, the functional approximation problem can be formulated as given a set of observations $\{(\vec{x}_k; y_k); k = 1, \ldots, n\}$ with $y_k = F(\vec{x}_k) \in \mathbb{R}$ and $\vec{x}_k \in \mathbb{R}^d$, it is desired to obtain a function $\mathcal{F}$ so $y_k \simeq \mathcal{F}(\vec{x}_k)$. Once this function is learned, it will be possible to generate new outputs from input data that were not specified in the original data set.

The reason to use RBFNNs [8] is because they have the capability of approximating any continuous function defined on a compact set. An RBFNN is a two-layer, fully connected network in which each neuron implements a Gaussian function. These kind of functions are very appropriate for function approximation because they are continuous, differentiable, provide a softer output, and improve the interpolation capabilities.

The real problem that arises when it is desired to approximate a function using an RBFNN is how to design the RBFNN. The parameters to be set to create an RBFNN are: the number and the position of the centers of the RBFs and their radii. The weights of the output layer can be calculated optimally solving a linear equation system. The solution space for the problem of initializing these variables is infinite since they are real values, on top of this, the risk of stalling in local minima is quite high.

The literature presents a wide variety of algorithms which are based on genetic algorithms (GAs) [2,21,47] and local search or descent gradient methods [26]. These techniques have shown a good performance, however in [36], memetic algorithms (MAs) were presented as evolutionary algorithms that hybridize the global optimization characteristics of GAs with local search techniques that allowed the GAs to perform a more deep exploitation of the solutions.

The two objectives of designing an RBFNN with the maximum generalization capabilities and the minimum approximation error with the training data is translated into defining the topology (number of RBFs) of the network. The more neurons are in the network, the smaller the approximation error will be although the more the generalization capabilities will be decreased. This fact

* Corresponding author.
E-mail address: aguillen@atc.ugr.es (A. Guillén).

0925-2312/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.neucom.2008.12.037

defines our task as a multiobjective problem which can be solved applying multiobjective GAs (MOGAs).

Another related issue is which variables the RBFNNs should consider. In real world applications there are many variables that can be useless. Having a number of irrelevant or redundant input variables can lead to overfitting, higher computational cost and to a poor generalization of the model. The algorithm presented in this paper evolves the input variables for the RBFNNs after a initialization based in the mutual information (MI) theory.

The algorithm presented in this paper will combine the techniques of local search, multiobjective optimization and mutual information system combined with pure variable selection through genetic algorithms to design an RBFNN that approximates a function with accuracy and generalization capabilities.

1.1. Description of RBFNN

Several types of artificial neural networks (ANNs) can be defined by modifying the activation function or the number of layers. In [8] Broomhead and Lowe introduced the radial basis function neural network model. This kind of ANN has been successfully applied to many problems related with non-linear regression and classification. Among the advantages of RBFNN over other types of networks, it can be noticed that the fast learning capabilities, the better interpretability of the model generated and ease of VLSI implementation [46]. An RBFNN is a two-layer, fully connected network in which each neuron implements a Gaussian function as follows:

$$\phi(\vec{x};\vec{c}_i,r_i)=\exp\left(\frac{-\|\vec{c}_i-\vec{x}\|^2}{r_i^2}\right),\quad i=1,\ldots,m, \tag{1}$$

where $\vec{x}$ is an input vector, $\vec{c}_i$ is the center and r_i is the radius of the i-th RBF.

The output layer implements a weighted sum of all the outputs from the hidden layer:

$$\mathscr{F}(\vec{x};C,R,\Omega)=\sum_{i=1}^{m}\phi(\vec{x};\vec{c}_i,r_i)\Omega_i, \tag{2}$$

where Ω_i are the output weights, which modulates the contribution of a hidden layer to the corresponding output unit and can be obtained optimally by solving a linear equation system.

1.2. Description of GA

Genetic algorithms were developed by Holland [29] in the 1970s and they are based on the evolutionary ideas of natural selection and genetics. The GAs generate a population starting from a previous one by crossing the individuals of the previous generation. This procedure allows the algorithm to exploit the historical information kept in the chromosomes of each individual in the population. Thus, if two good individuals are crossed, it is quite probable that the resulting offsprings improve the solution of the problem to be solved. In classical GAs, each individual in the population encodes a solution to the problem using a chromosome composed by a sequence of genes whose values are 0 or 1.

The general scheme that classical GAs follow is:

```
randomly initialize population(t)
determine fitness of population(t)
do
    select parents from population(t)
    perform crossover on parents to generate population(t+1)
    mutate population(t+1)
    compute fitness of population(t+1)
while termination criterion
```

2. A parallel evolutionary feature selector and RBFNN designer for function approximation: pEFSFA

This section describes the proposed algorithm that optimizes the inputs, the structure and the parameters of RBFNNs for function approximation problems. This algorithm implements and combines several paradigms such as MAs, MOGAs and PGAs. The synergy of these techniques results in a robust algorithm which is able to design proper RBFNNs.

2.1. Representing RBFNNs in the individuals

As it was shown in the Introduction, to design an RBFNN it is needed to specify: (1) the variables that the RBFNN will receive as inputs, (2) the number of RBFs, (3) the position of the centers of the RBFs, (4) the length of the radii and (5) the weights for the output layer.

An individual will encode, as a binary vector, the input variables that will take of each input vector, then, the position of the centers in that input variable space and the radii are stored as real numbers. The binary encoding was chosen because of its simplicity and its discrete solution space.

2.2. Initial population

The infinite solution space for the problem of setting the centers and the radii and the large (although it depends on the problem) solution space for the problem of selecting the input variables, makes very important the initialization of the population in areas of the solution space where the solutions can be considered adequate. Therefore, this subsection deals with the method used to initialize the individuals.

2.2.1. Mutual information systems

Since the individuals represent the set of input variables using a binary vector, the number of possible solutions is 2^d. At first, this might not seem too high but, to evaluate the goodness of each one of those possible solutions, infinite RBFNNs could be designed. Therefore, a preprocessing of these input variables that indicates which variables are the most significant for the output becomes necessary. For this purpose, the mutual information (also called cross-entropy) concept will be employed. Let $X^l=\{\vec{x}_k^l\}$ with $l\in 1,\ldots,d$ (i.e. X^l is the l-th input variable) and $Y=\{y_k\}$ with $\{k=1,\ldots,n\}$ then, the mutual information between X^l and Y can be defined as the amount of information that X^l provide about Y, and can be expressed as

$$I(X^l,Y)=H(Y)-H(Y|X^l), \tag{3}$$

where $H(Y)$ is the entropy of variable Y that measures the uncertainty on Y. In the continuous case and according to the formulation of Shannon, it is defined as

$$H(Y)=-\int\mu_Y(y)\log\mu_Y(y)\,dy, \tag{4}$$

where $\mu_Y(y)$ is the marginal density function, that can be defined using the joint probability density function (PDF) $\mu_{X^l,Y}$ of X^l and Y as

$$\mu_Y(y)=\int\mu_{X^l,Y}(x,y)\,dx \tag{5}$$

and $H(Y|X^l)$ is the conditional entropy that measures the uncertainty of Y, given that X^l is known. $H(Y|X^l)$ is defined in the continuous case as

$$-\int\mu_X^l(x)\int\mu_Y(y|X^l=x)\log\mu_Y(y|X^l=x)\,dy\,dx. \tag{6}$$