Chemometrics and Intelligent Laboratory Systems 124 (2013) 50–57

Contents lists available at SciVerse ScienceDirect

Chemometrics and Intelligent Laboratory Systems

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

Adaptive configuration of radial basis function network by regression tree allied with hybrid particle swarm optimization algorithm

Rui-Min Luo [a], Ya-Qiong Li [a], Hai-Li Guo [a], Yan-Ping Zhou [a,*], Hui Xu [a], Hong Gong [b,*]

[a] Key Laboratory of Pesticide and Chemical Biology of Ministry of Education, College of Chemistry, Central China Normal University, Wuhan 430079, PR China
[b] Economics and Management School, Wuhan University, Wuhan 430072, PR China

ARTICLE INFO

Article history:
Received 12 October 2012
Received in revised form 22 January 2013
Accepted 6 February 2013
Available online 16 February 2013

Keywords:
QSAR
Radial basis function network
Regression tree
Particle swarm optimization
1-[(2-Hydroxyethoxy)methyl]-6-(phenylthio)thymine (HEPT) analogues
Flavonoid derivatives

ABSTRACT

Configuration of a radial basis function network (RBFN) comprises identifying the network parameters (inputs, centers as well as widths in RBF units, and weights between the hidden and output layers) and architecture. The issues of overfitting and local optima often happened during RBFN training. To rectify this situation, regression tree (RT), allied with hybrid particle swarm optimization (PSO) algorithm, was invoked to configure an RBFN to form the HPSORTRBFN algorithm in the present study. Discrete PSO was invoked to obtain an RT of the right size. The regions in the instance space defined by the leaf nodes of the grown RT were transformed into the centers in RBF units and the number of leaf nodes acted as the network structure. The splitting variables in RT became the inputs in RBFN. The widths and weights in RBFN were simultaneously optimized by continuous PSO. HPSORTRBFN was applied to predict the anti-HIV activities of 1-[(2-Hydroxyethoxy)methyl]-6-(phenylthio)thymine (HEPT) analogues and the bioactivities of flavonoid derivatives. The results showed RT allied with HPSO is able to configure a globally optimal RBFN and HPSORTRBFN owns superior modeling performance to RBFN and RT.

1. Introduction

Radial basis function network as an important non-parameter modeling tool has been proven fruitful in quantitative structure–activity relationship (QSAR) studies. It may be due to the fact that RBFN holds great potential in approximating various nonlinear relationships between the descriptors and bioactivities within sufficient accuracy [1]. This is also the right advantage of RBFN over other methods, e.g., partial least-squares (PLS) regression.

The design of a successful RBFN involves several nontrivial issues, such as the network architecture and parameters (inputs, centers and widths in RBF units, and weights between the hidden and output layers) [2]. Typically, RBFN is configured by first identifying the centers and widths via clustering method (i.e., K-means) under a fixed network topology [3,4] and then obtaining the weights by a regularization method or descent algorithm [5]. Though the clustering method is able to allocate efficaciously the training instances into clusters, it only takes the inputs into consideration and pays no attention to the outputs. Imaginably, the followed network weight calculation by regularization method may yield underfitting and suboptima. The descent algorithm offers the possibility of improving the centers and widths, however, suffering from some well-known drawbacks, e.g., premature convergence to suboptima and inclination to overfitting. The RBFN topology, i.e., the hidden node number, is generally pre-defined. As a matter of fact, it is one of the most crucial factors for RBFN to effectively solve problems. Oversimplified network might hamper the network convergence, while excessive hidden nodes would be prone to overfit the data, hence offering poor generalization. This is a manifestation of the ubiquitous issue of model complexity encountered by all non-parameter methods. Several tools are used for the network structure identification by gradually adding or deleting the hidden nodes [6–8]. However, the performance of such methods depends strongly on the concrete operation of the network because the architecture is not fixed preliminarily but evolved during regulating the whole system. In addition, RBFN must be coupled with another feature selection tool for identifying the inputs, otherwise yielding deteriorated performance and prohibitively large computation.

Owing to the great appeal of RBFN, many efforts were made to meliorate RBFN. For example, optimization techniques, such as, particle swarm optimization (PSO) [9] and genetic algorithm [10], were attempted for inducing the globally optimal RBFNs. Although the generalization ability of RBFN is indeed enhanced to a great extent, such optimization tools confine the centers initialized in the first generation to the random subset of the training set or the randomness without any apriority on the instances. Such limited feasible regions for centers may not provide a sufficiently flexible model to guarantee acceptable performance. Little attention was paid on inputs which were usually pre-specified before configuring RBFN. In addition, regression tree (RT) was explored by Orr [11] to initialize RBFN. In Orr's study [11], subset of the leaf nodes in an unpruned RT contributed all RBF

* Corresponding authors. Tel.: +86 15872406428; fax: +86 27 67867141.
E-mail addresses: hgzyp2005@yahoo.com.cn (Y.-P. Zhou), Gonghong009@yahoo.com.cn (H. Gong).

0169-7439/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.chemolab.2013.02.002

units. The splitting variables in RT acted as the inputs in RBFN so as to improve the computational effectivity. Subsequently, traditional regularization method (i.e., pesudoinverse matrix) was used to calculate the weights between the hidden and output layers. This may be due to the tree-induction process decomposing the instance space gradually into relatively pure disjoint regions and the natural feature of RT in automatically deciding the relevant variables. The initialization of RBFN by RT shows good theoretical and experimental performance [11]. However, the subset of leaf nodes in RT by forward selection method is computationally expensive with less intuitiveness, since the selection is processed by checking each node in RT from root to leaves [11]. Only part of leaves is ultimately involved for contributing RBF units, thus some regions will be missed, resulting in an increased risk of RBFN being trapped into suboptima. Moreover, RBFN gets into local optima with a higher frequency, since the tree induction and RBFN configuration are two sequential and completely independent processes. Consequently, designing a successful RBFN involves several nontrivial issues, and so far there does not seem to exist any simple and general algorithm or heuristic addressing them all at the same time.

Inspired by the attractive characteristic of RT in initializing RBFN, the appealing property of PSO in parameter optimization, and the requirements of RBFN induction, RT allied with hybrid PSO was invoked to configure an RBFN (HPSORTRBFN) via simultaneously identifying the parameters and topology in RBFN in the current study. In HPSORTRBFN, the appropriately-fit tree was obtained by discrete PSO, with the splitting variables serving as the inputs of RBFN and each leaf node deciding one center in RBFN. The remainder parameters in RBFN, i.e., the widths and weights, were optimized by continuous PSO. Two QSAR data sets were used to support the viewpoint that RT allied with hybrid PSO serves as the efficacious way to improve the generalization ability of RBFN.

2. Theory

2.1. Radial basis function network (RBFN)

RBFN consists of three layers: input, hidden, and output layers. The input layer merely acts as the input distributor to the hidden layer. Each hidden node takes a basis function as nonlinear transfer to operate on the inputs. The output layer is operated by a linear combination of RBF units according to the following expression:

$$y_k(\mathbf{x}) = \sum_{j}^{J} w_{kj}\phi_j(\mathbf{x}) + w_k \tag{1}$$

where y_k is the kth output unit for the input vector $\mathbf{x}$, J is the number of RBF units, w_{kj} is the weight between the kth output and the jth hidden nodes, ϕ_j is the notation for the output of the jth RBF unit and w_k is the bias. The most commonly used RBF is the Gaussian function characterized by a center $\mathbf{c}_j$ and a width σ_j as follows:

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\left(\mathbf{x}-\mathbf{c}_j\right)^{\mathrm{T}}\left(\mathbf{x}-\mathbf{c}_j\right)}{\sigma_j^2}\right) \tag{2}$$

From the above-mentioned delineation, one can obtain that the core of RBFN configuration comprises the identification of the network parameters (inputs, centers, widths, and weights) and architecture. In this paper, RT incorporated with hybrid PSO was implemented for adaptively configuring RBFN by simultaneously seeking for the optimal network parameters and architecture.

2.2. Regression tree (RT)

RT is originated from classification and regression trees (CART) by Breiman et al. [12]. Here only a concise description of RT is presented.

Generally, the configuration of RT consists of three basic steps. Firstly, the largest tree is grown by applying greedy recursive partitioning. Recursive partitioning is conducted in top-down fashion, starting from the root node containing the entire training compounds until each node reaches complete homogeneity or minimal sample number (i.e., node size) and becomes a terminal or leaf node. In essence, the tree-induction process, i.e., recursive partitioning, decomposes the instance space gradually into relatively pure disjoint regions. The partition proceeds via selecting a certain descriptor and its certain value, respectively, as the splitting variable and value on a goodness-of-split criterion [12]. Typically, the descriptors carrying most information about the endpoint tend to be split earliest and most often. This is some kind of automatic relevance determination of variables which is a natural feature of RT. Secondly, on the minimal cost-complexity pruning (MCCP) criterion [12], the largest tree is pruned to yield a sequence of nested subtrees. Ultimately, from such nested subtrees, the final appropriately-fit RT is selected [12]. In the current study, RT was used to identify the inputs, centers, and architecture corresponding to RBFN. Since the size of RT does not predicate the complexity of RBFN, it is not necessary to perform the pruning step that is normally associated with recursive splitting methods. Therefore, the minimal node size (i.e., p_{min}), is the only tunable parameter to decide when to stop growing RT. Here, p_{min} is optimized by discrete PSO.

2.3. Particle swarm optimization (PSO)

The particle swarm optimization method, originally developed by Eberhart and Kennedy [13–16], is a stochastic global optimization method simulating the social behavior of bird flock. It explores the problem space by a population of particles, each standing for a single solution. In PSO, each particle flies over the problem space with a velocity guiding the flying of the particle, keeping track of the best solution encountered so far. PSO can operate in continuous and discrete spaces, being indicated fast convergence to the optima [9,17–20]. The detailed description of PSO can be found elsewhere [9]. Here PSO is briefly described.

For continuous PSO, position and velocity of each particle is first initialized by dispersing them uniformly across the search space randomly. The ith particle and its corresponding velocity, i.e. the rate of the position change for the ith particle, are represented as $\mathbf{Par}_i = (Par_{i1}, Par_{i2}, \ldots, Par_{iD})$ and $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, respectively. In every cycle, updating each particle is realized by following the personal best position and the global best position. The former refers to the best previous position of the ith particle yielding the best fitness value, represented as $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, while the latter is the best particle among all the particles in the population, represented as $\mathbf{p}_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$. Once the above-mentioned two best values have been found, the particle updates its velocity and position in terms of the following two equations:

$$v_{id}(\text{new}) = v_{id}(\text{old}) + c_1 * r_1 * (p_{id} - Par_{id}) + c_2 * r_2 * \left(p_{gd} - Par_{id}\right) \tag{3}$$

$$Par_{id}(\text{new}) = Par_{id}(\text{old}) + \mu * v_{id}(\text{new}) \tag{4}$$

where c_1 and c_2 are two positive constants named learning factors, taking the integer value 2; r_1 and r_2 are random numbers in the interval (0, 1). In Eq. (4), μ, a random number uniformly distributed in (0, 1), is the restriction factor to determine velocity weight. The particle's velocity is renovated by employing Eq. (3) according to its previous velocity and the distances of its current position from its