Neural Comput & Applic (2013) 22:1655–1666
DOI 10.1007/s00521-012-1053-8

ORIGINAL ARTICLE

Reviewing and designing pre-processing units for RBF networks: initial structure identification and coarse-tuning of free parameters

Gökhan Kayhan · Ali Ekber Ozdemir · İlyas Eminoglu

Received: 26 April 2011 / Accepted: 22 June 2012 / Published online: 10 July 2012

Abstract This paper reviews some frequently used methods to initialize an radial basis function (RBF) network and presents systematic design procedures for pre-processing unit(s) to initialize RBF network from available input–output data sets. The pre-processing units are computationally hybrid two-step training algorithms that can be named as (1) construction of initial structure and (2) coarse-tuning of free parameters. The first step, the number, and the locations of the initial centers of RBF network can be determined. Thus, an orthogonal least squares algorithm and a modified counter propagation network can be employed for this purpose. In the second step, a coarse-tuning of free parameters is achieved by using clustering procedures. Thus, the Gustafson–Kessel and the fuzzy C-means clustering methods are evaluated for the coarse-tuning. The first two-step behaves like a pre-processing unit for the last stage (or fine-tuning stage—a gradient descent algorithm). The initialization ability of the proposed four pre-processing units (modular combination of the existing methods) is compared with three non-linear benchmarks in terms of root mean square errors. Finally, the proposed hybrid pre-processing units may initialize a fairly accurate, IF–THEN-wise readable initial model automatically and efficiently with a minimum user inference.

G. Kayhan (✉)
Computer Engineering Department, Ondokuz Mayıs University, Samsun, Turkey
e-mail: gkayhan@omu.edu.tr

A. E. Ozdemir
Unye Meslek Yuksek Okulu, Ordu University, Ordu, Turkey

İ. Eminoglu
Electrical and Electronics Engineering Department, Ondokuz Mayıs University, Samsun, Turkey

Keywords Counter propagation network (CPN) · Fuzzy C-means (FCM) · Gustafson–Kessel (GK) · Radial basis function (RBF) · Hybrid training and modeling · Partition validations

1 Introduction

Radial basis function (RBF) neural networks have been successfully applied for nonlinear function approximation and data classification in a wide range of areas, such as signal processing, system modeling, control, and fault diagnosis. Simple structure of RBF enables learning in stages and gives a reduction in the training time, and this has lead to the application of such networks to many practical problems. The adjustable parameters of such networks are the receptive field centers (the location of basis functions), the width (the spread), the shape of the receptive field, and the linear output weights. The learning strategies in the literature used for the design of RBF network differ from each other mainly in the determination of centers. These can be categorized into the following groups [1]:

1. Fixed centers assigned randomly among input samples: In this method, which is the simplest one, the centers are chosen randomly from the set of input training samples. As a natural consequence, without any prior knowledge about the prototype vectors, the number of centers to represent the data would be large and constructed RBF network will not be optimal in size.
2. Orthogonalization of regressors: The common feature of this class of algorithms is that they orthogonalize the regressors to decouple the contribution of each regressor to an energy-based performance index.

The most commonly used algorithm is OLS [2], which selects a suitable set of centers (regressors) but might not be the optimal set as demonstrated in [3] among input training samples. The procedure chooses centers one by one, so that, at each step, the selected center maximizes the increment to the output energy.

3. Supervised selection of centers: In this method, the centers together with all other parameters of RBF network (linear weights, variances) are updated using a back-propagation-like gradient descent algorithm. The main disadvantage of this approach is that the selection of initial structure and associated parameters are either in random or trial–error fashion. This may lead to under or over-sized networks with poor initial parameters and unsatisfactory performances.
4. Input clustering (IC) [4]: The locations of centers are determined by a clustering algorithm applied to input training sample vectors. In the literature, different clustering algorithms have been used for center determination of RBF network: (1) because real data vary considerably, it is difficult for a clustering algorithm to fit all real cases, (2) the number of cluster is to be known in prior.
5. Input output clustering (IOC): The IC method in (4) is based on the distribution of the training inputs alone and does not take into consideration the output values that do influence positioning of the centers, especially when variation of output in a cluster is high. So, centers are also selected based on both input and output data or joint input–output data such as [1].
6. Evolutionary algorithms: All RBF parameters including centers (and also the number of centers), widths, and linear weights are optimized by genetic algorithms according to defined (single or multi-objective) cost function [5–7], but this approach can be computationally expensive.

There are many works on determining the centers of RBF networks in literature. In [8], a novel learning algorithm merging the automatic model selection criterion that uses the Bayesian information criterion (BIC) method to select the appropriate number of basis function for OLS and a new method to optimize the widths of Gaussian functions that improves the generalization performance are proposed. A multi-output fast recursive algorithm (MFRA) that is used for selecting the centers of multi-output RBF network and estimating the network weights is proposed in [9]. In order to develop two-step learning algorithm, two learning algorithms are combined. In unsupervised step, the input data are analyzed by using the self-organizing map (SOM). In supervised step, some snaptic weights of SOM are chosen and then the selected snaptic weights are used as centers of hidden neurons of the RBF network [10].

Several heuristic hybrid learning methods, which apply a clustering algorithm for locating the centers and subsequently a linear least squares method for the linear weights, have been previously suggested with considerable success for many applications. In [11] to train Gaussian-type RBF neural networks, a novel clustering-based algorithm is introduced, and a specialized learning strategy that combines the fuzzy and crisp clustering is developed. The produced hybrid clusters are used to estimate the neural network's parameters such as centers and widths. In [12], there is a three-phase learning algorithm that optimizes the functionality of the optimum steepest descent (OSD) learning method. In the first phase, K-Means Clustering algorithm is used for adjusting centers of the RBF layer, and in the second phase, *p*-nearest neighbor algorithms are used for calculating center widths. In the third phase, the output weights are determined by the OSD method. An improved performance of the RBF network trained in the hidden layer with conditional fuzzy clustering (CFC) with respect to classical FCM is given in [13]. But, CFC has one drawback that one or more appropriate linguistic contexts over the output space, expressed in the form of fuzzy set, must be built by first observing the distribution of the values in the output space. In [14], a novel clustering technique improves the performance obtained with CFC; furthermore, it eliminates the drawback of CFC [15] and further reduces the error rates with respect to [13, 14]. A multi scale RBF network is proposed unlike a conventional standard single scale RBF network in [16]. A K-Means Clustering algorithm and an improved orthogonal least squares (OLS) algorithm are used to determine the unknown parameters in the network model including the centers and widths of the basis functions, and the weights between the basis functions.

Combining advantages of existing methods might potentially lead to some hybrid training methods such as [5, 6, 17–20].

This paper suggests two-stage pre-processing units for constructing RBF network in a systematic way with a minimum user inference along with a good interpretability capacity. This paper is an extension of preliminary results given in [21, 22]. The general framework of proposed hybrid three-stage structures (a two-step pre-processing unit(s) and a fine-tuning stage) is shown in Fig. 1. The last step (fine tuning of parameters) is also added to evaluate the performance of pre-processing units.

Each step of Fig. 1 has a unique operational target and contributes model construction in a sequential manner. These three stages are summarized as below:

In the first step of the hybrid algorithm, the number and the locations of the centers of RBF network are determined using either the OLS algorithm with user-defined (ε_{OLS}) termination parameter or the modified counter propagation

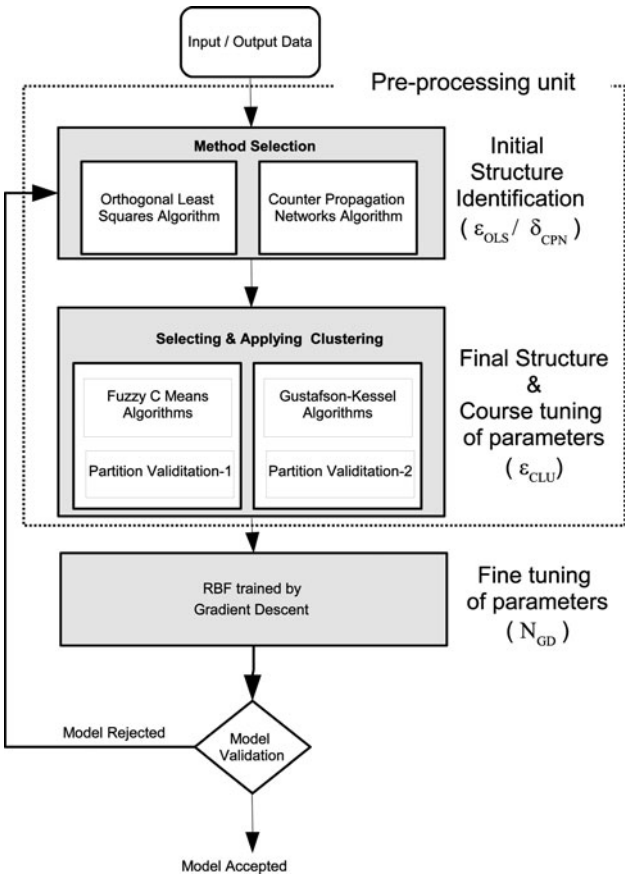

Fig. 1 General framework of the proposed three-stage (preprocessing unit(s) and fine-tuning stage) hybrid algorithms

network (CPN) with user-defined cluster radius (δ_{CPN}) parameter. A total of M initial centers represented by $C = \{c\ (x_1, y_1), \ldots, c\ (x_M, y_M)\}$ are generated either by of the algorithms (OLS or CPN). Weight vector of the output layer is assigned as $W = \{w_1 = y_1, \ldots, w_M = y_M\}$ and passed into the next step. These M centers also represent M distinct clusters, but locations of clusters and associating weights may not be optimal.

In the second step, a coarse-tuning of free parameters of the RBF network (centers, widths, and weights) is achieved by using the clustering procedures. These are either the GK or the FCM with a pre-defined ε_{CLU}; the partition validation algorithms embedded into the clustering algorithms may further reduce the number of centers since M found in the first stage may not be optimal. These two stages construct pre-processing units.

The third step is so-called a fine-tuning stage. The coarsely tuned RBF network is optimized by using the gradient descent (GD) algorithm (with a user-defined N_{GD}). Then, the final form of RBF network can be formed to evaluate in a way that one may decide which pre-processing unit can produce the lowest root mean square (RMS) errors.

Depending on which pre-processing algorithms utilized on the first two steps, four pre-processing units are formed namely (1) (OLS + GK), (2) (CPN + GK), (3) (OLS + FCM), and (4) (CPN + FCM). Structural and parametric initializations of RBF network are achieved by the combinations of well-known algorithms in harmonies ways in the first two-step of the hybrid structure. To evaluate the performance of the proposed pre-processing units, each pre-processed structure is optimized by using a gradient descent (GD) type algorithm. The structure of the last method (CPN + FCM) is initially proposed in [23] to construct a normalized fuzzy system for model construction. A modified CPN is exploited as a pre-processor to extract a number of clusters that can be viewed as an initial fuzzy model from the raw data [23, 24]. The fine-tuning step is achieved by using a back-propagation type of learning.

The proposed pre-processing units are tested over three non-linear synthetic input–output data sets in terms of RMS errors. The proposed methods have been compared with similar (if possible an identical) number of sub-clusters. Hence, the effect of structural and parametric initialization of RBF network is demonstrated without bias. Readability of constructed model is an important aspect for end-user. Each Gaussian basis function of RBF can easily be represented by fuzzy IF–THEN rule. For example, one of the constructed RBF network is represented by equivalent IF–THEN rules. Hence, the interpretation capability of constructed models allows domain experts to relate, discuss, and present local cause-effect dynamics of physical system and corresponding local IF–THEN rule.

This paper is organized as follows. General introduction material to RBF network is given in Sect. 2. Various learning algorithms are including GD, OLS, CPN, FCM clustering, GK clustering algorithm, and finally partition validation procedures discussed given in Sect. 3. In Sect. 4, the proposed pre-processing units are tested in terms of structural and parametric initializations of RBF model and results are presented. The obtained results and proposed pre-processing units have been comparatively discussed in Sect. 5. Finally, conclusions and future works are outlined in Sect. 6.

2 Radial basis function network

A multi-input and single-output RBF network consisting of M hidden nodes (units or basis functions) is shown in Fig. 2. Each hidden unit converts a multi-dimensional input vector into a single-dimensional output vector. From the computational viewpoint, RBF network can be seen as a set of weighted basis functions. Hidden units can be made of various types of activation functions such as the

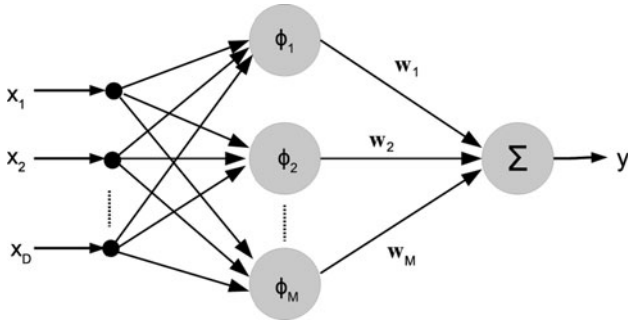

Fig. 2 A multi-input and single-output RBF network

thin-plate-spline, the multi-quadratic, the inverse multi-quadratic, and the Gaussian. The most common choice is the Gaussian type, and it is adopted in this work. The functional equivalence between RBF and fuzzy inference system under minor assumptions is shown in [25–28].

In Fig. 2, x_D represents D-dimensional input vector applied to RBF network, Φ_j is the output of jth activation function, w_j is the weight vector of jth output layer, and output of RBF is represented by y. M is number of the maximum basis function (Φ) that is used in RBF network; w_M is the weight vector of Mth basis function. The output of jth basis function is calculated as shown in (1) where c_j represents centers and σ_j represents widths of the Gaussian function. The overall output of RBF network is then given in (2). The output can also be given in (3) as a general form. The term of $\|x - c_j\|$ appearing in (1) and (2) refers to measuring the Euclidean distance between input vector and center of jth hidden unit [29].

$$\Phi_j = \mathrm{e}^{-\frac{\|x-c_j\|^2}{\sigma_j^2}} \tag{1}$$

$$y = \sum_{j=1}^{M} w_j \Phi_j \tag{2}$$

$$y = \sum_{j=1}^{M} w_j \mathrm{e}^{-\frac{\|x-c_j\|^2}{\sigma_j^2}} \tag{3}$$

3 Learning algorithms

In this section, two well-known algorithms (OLS and CPN) are reviewed, each algorithm can be potentially be used to determine initial structure of RBF network. CPN is easy and straightforward as compare to OLS algorithm. Final structure and coarse-tuning of RBF can be handled with clustering procedure. Thus, FCM and GK clustering algorithms are given in details (along with partition validations procedures to decide the final number of clusters). As shown in Fig. 1, the fine-tuning step is equipped with GD algorithm for the sake of simplicity. A brief treatment of the GD method is given below.

3.1 Gradient descent (GD) algorithm

The GD algorithm utilizes a cost function given in (4) and detailed treatment of the method can be found in [30]. The desired output of RBF network is represented by $d(n)$, actual output is $y(n)$, and L shows total number of input data. Input–output data set is applied during training N_{GD} times (the number of iteration), and the main goal is to minimize total cost function given in (5).

$$E = \frac{1}{2}\sum_{n=1}^{L} (d(n) - y(n))^2 \tag{4}$$

$$T = \min\left(\sum_{i=1}^{N_{\mathrm{GD}}} E_i\right) \tag{5}$$

The free parameters of RBF network (widths, centers, and weights) using GD algorithm can be computed using (6).

$$\Phi^{(n+1)} = \Phi^{(n)} - \mu\frac{\mathrm{d}E^{(n)}}{\mathrm{d}\Phi^{(n)}} \tag{6}$$

3.2 Sub-clusters using orthogonal least squares (OLS) algorithm

The orthogonal least squares (OLS) algorithm is one of the most popular procedures for the training of radial basis function networks (RBF network) [2]. The OLS algorithm is a greedy algorithm that chooses RBF centers one by one from existing training data in a systematic way according to individual contribution to error reduction [31]. The OLS method has superior numerical properties compared with the ordinary least squares (LS) method. Our interest in OLS method, however, is to use it for subset (cluster) selection. The major disadvantage of using OLS is that the RBF widths are set a priori and the centers of the RBF nodes are selected from a set of training samples (with limited size), the optimal values on the continuous parameter space for the center and width parameters of RBF nodes may not be optimal.

$d = [d(1)d(2)\cdots d(L)]^{\mathrm{T}}$	desired output vector
$\Omega = [\Phi_1\Phi_2\cdots\Phi_M]$	regression matrix and
$\Phi_i = [\Phi_i(1)\,\Phi_i(2)\cdots\Phi_i(L)]^{\mathrm{T}}$	output of hidden units
$W = [w_1 w_2 \cdots w_M]^{\mathrm{T}}$	weight of output layer
$E = [\varepsilon(1)\varepsilon(2)\cdots\varepsilon(L)]^{\mathrm{T}}$	error vector
$Q = [q_1 q_2 \cdots q_M]$	represent orthogonal vector

OLS algorithm can now be given in compact form as below:

1. The following computations are to be carried out for all center candidates and $(1 \leq i \leq M)$

$q_1^i = \Phi_i$ Setorthogonalvectors

$g_1^i = \frac{(q_1^i)^{\mathrm{T}} d}{(q_1^i)^{\mathrm{T}} q_1^i}$ CalculateLSsolution

$[r]_1^i = \frac{(g_1^i)^2 (q_1^i)^{\mathrm{T}} q_1^i}{d^{\mathrm{T}} d}$ Calculaterelativeerrorrateforeachq_1^i

$q_1 = \max\left[\frac{(g_1^i)^2 (q_1^i)^{\mathrm{T}} q_1^i}{d^{\mathrm{T}} d}\right]$ Selectfirstorthogonalvector

2. For kth step $(k \geq 2)$, for $(1 \leq i \leq M)$ and following computations are to be done.

$\alpha_{jk}^i = \frac{q_j^{\mathrm{T}} \Phi_i}{q_j^{\mathrm{T}} q_j}, \quad 1 \leq j < k$

$q_k^i = \Phi_i - \sum_{j=1}^{k-1} \alpha_{jk}^i q_j$ Set orthogonal vectors

$g_k^i = \frac{(q_k^i)^{\mathrm{T}} d}{(q_k^i)^{\mathrm{T}} (q_k^i)}$ Calculate LS solution

$[r]_k^i = \frac{(g_k^i)^2 (q_k^i)^{\mathrm{T}} (q_k^i)}{d^{\mathrm{T}} d}$ Calculate relative error rate for each q_k^i

Similarly, if this regressor $q_k^i = \max\left[\frac{(g_k^i)^2 (q_k^i)^{\mathrm{T}} (q_k^i)}{d^{\mathrm{T}} d}\right]$ is kept out from being a regressor then the largest error can be generated. In each iteration, previously selected centers are excluded and remaining centers are taken into account to select new centers. Selection procedure is terminated in M_s step as follow:

$$1 - \sum_{j=1}^{M_s} [r]_j < \varepsilon_{\mathrm{OLS}} \tag{7}$$

And $\varepsilon_{\mathrm{OLS}}$ is selected between of 0 and 1. Classic Gram–Schmidt orthogonality method is explained and employed in this work. In literature, modified Gram–Schmidt method and Householder transformation are also used for orthogonalization [2].

3.3 Generating sub-clusters using modified counter propagation network (CPN)

The CPN is of a very simple structure. The trained CPN functions as a look-up table. This section introduces the CPN network training.

3.3.1 The Kohonen layer

The self-organizing Kohonen network in Fig. 3 is introduced to produce the sub-clusters [24, 32]. In contrast to the Kohonen network [33], modified Kohonen network has a variable structure in which the number of nodes in the competitive layer can be changed dynamically in response to the incoming data. Starting with zero node at the

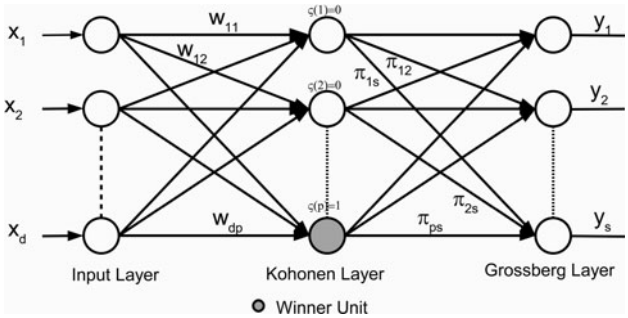

Fig. 3 Counter propagation network

competitive layer, as the learning process proceeds with increasing iteration, the number of the nodes grows accordingly. The number of clusters can mainly be controlled by the cluster radius (δ_{CPN}). Depending on the value of the cluster radius (δ_{CPN}), the number of sub-clusters varies from only 1 to L. It simply means that if (δ_{CPN}) is selected as large as to cover all data points within the cluster radius then only one sub-cluster (or node) can be found. Inversely, if δ_{CPN} is selected very small then every data point can be a center of a sub-cluster and the number of sub-clusters can be equal to the number of input data or L. The activation threshold (Ns) shows how many times a particular node is being activated by the input vector for a constant value of (δ_{CPN}). The activation threshold (Ns) and the first incoming data (ψ_1) for each new node also affect the training of CPN.

The procedure for the modified competitive learning algorithm is described as follows [32].

Step 1: Initialization.
Consider a set of L data points $P_1, P_2, \ldots, P_L$, the multi-input and multi-output I/O data pair can be represented as $P_k = (x_{k1}, x_{k2}, \ldots, x_{kd}, y_{k1}, y_{k2}, \ldots, y_{ks})$ $k = 1, 2, \ldots, L$. As can be seen from Fig. 3a, d-dimensional input vector is fed into the input layer and the current input set is denoted by $\psi = [\psi_1, \psi_2, \ldots \psi_L]$ where $\psi_k = (x_{k1}, x_{k2}, \ldots, x_{kd})$ and $Y = [Y_1, Y_2, \ldots, Y_L]$ where $Y_k = (y_{k1}, y_{k2}, \ldots, y_{ks})$. Specify the valid radius (δ_{CPN}) for all nodes. Set the number of nodes $N_t = 1$ and the activation number of node 1 $N_{s1} = 1$ and set the iteration number $l = 1$. Let the first weight vector w_1 be the first input pattern ψ_1, as $w_1 = \psi_1$.

Step 2: Similarity calculation.
For the lth input pattern, find the node p that has the minimum distance D to the current input pattern ψ_1 by.

$$D = \min\left\|w_p(l) - \psi_1(l)\right\| \quad p = 1 \ldots N_t \tag{8}$$

The Euclidean distance is defined as

$$\left\|w_p - \psi_l\right\|^2 = (x_p - \psi_i)(x_p - \psi_i)^{\mathrm{T}} \tag{9}$$

Step 3: Determine the winning node and update weight vector. Use the following rule:

1. If $\|w_p(l) - \psi_1(l)\| < \delta_{\text{CPN}}$ then node p is winner and the output of node p is $\varsigma(p) = 1$ and modify the weight vector of p to

$$w_p(t) = w_p(t-1) + \alpha(\psi_1 - w_p(t-1))\varsigma(p) \tag{10}$$

where $0 < \alpha < 1$ is a gain sequence decreasing monotonically with the activation threshold of the winner node and update the rest of the parameters as follows:

$$N_s(p) = N_s(p) + 1 \tag{11}$$

$$\alpha = \alpha_0 / N_s(p) \tag{12}$$

$$l = l + 1 \tag{13}$$

2. If $\|w_p(l) - \psi_1(l)\| > \delta_{\text{CPN}}$ then create a new node. Because the incoming input data have a greater distance than δ_{CPN} to all existing nodes, none of the nodes can accommodate the data and a new node is required. The rest of the parameters are updated as follows:

$$w(p) = \psi(l) \tag{14}$$

$$N_s(p) = 1 \tag{15}$$

$$p = p + 1 \tag{16}$$

If $l < L$, go to Step 2, otherwise set $p = N_t$ and stop.

3.3.2 Grossberg layer

The CPN training algorithm is a supervised training process in response to a set of paired training samples (ψ_k, Y_k). The algorithm essentially consists of two parts: a Kohonen scheme that is unsupervised in nature and is used to learn w^j and a Grossberg scheme that is truly supervised and is used to learn π^j.

Once the Kohonen layer has stabilized, w^j can be frozen. Then, the Grossberg layer begins to learn the desired output Y_k for each frozen weight vector by adjusting the weights (π^j) that connecting the Kohonen units to the Grossberg units. More precisely, the update law at this layer is given by

$$\pi_g^j(t) = \pi_g^j(t-1) + \beta\left[-\pi_g^j(t-1) + Y_k\right]\varsigma(p) \tag{17}$$

π_g^j where is a connection weight from the jth Kohonen unit to the gth Grossberg (output) unit. β is a constant update rate within the range [0, 1], Y_k is the kth component (the desired output) of the training sample Y. And $\varsigma(p)$ is the output of the winner node in the Kohonen layer. More specifically, a trained CPN network has been arranged in such a way that P associations (w_j, π^j) can function as a look-up table. The look-up table might be seen as a set of fuzzy IF w_j THEN π^j rules. In terms of knowledge representation, CPN can be thought of a hard version of fuzzy representation [32] and each rule represents one sub-cluster.

3.4 Fuzzy C-means (FCM)

Most analytical fuzzy clustering algorithms are based on optimization of the basic C-means objective function or some modification of it. The most widely used clustering algorithm is the FCM due to its efficacy and simplicity. However, the number of clusters c is required to be predetermined. FCM algorithm partitions a collection of L data points ($X = \{x_1, x_2, \ldots, x_L\}$) into c fuzzy clusters such that the objective function is minimized, where m is a fuzzy coefficient, V_i is the prototype of the ith cluster generated by fuzzy clustering, U_{ik} (fuzzy partition matrix) is the membership degree of the kth data belonging to the ith cluster represented by V_i. The standard Euclidean norm is formed when A_i in (19) selected as unity matrix $(A_i = I)$

$$J(X, U, V, A) = \sum_{i=1}^{c}\sum_{k=1}^{L}(U_{ik})^m D_{ikAi}^2 \tag{18}$$

$$D_{ikAi}^2 = (x_k - V_i)^{\mathrm{T}} A_i (x_k - V_i) \tag{19}$$

The measure of dissimilarity in (19) is squared distance between each data point (x_k) and the cluster prototype (V_i). The squared distance is further weighted by power of the membership degree of that point $(U_{ik})^m$. The value of the cost function (18) can be seen as a measure of the total variance of x_k from V_i and the fuzzy C-means algorithm (FCM) is given below [33]:

Step 1: Given the data set X, choose the number of cluster $(2 < c < L)$, the weighting exponent $(m > 1)$, the termination tolerance $(\varepsilon_{\text{CLU}})$, and norm-inducing matrix $(A_i = I)$. Initialize the partition matrix randomly (0–1) in (20). $m = 2$ is used in this work.

$$\sum_{i=1}^{c} U_{ij} = 1 \quad j = 1, \ldots, L \tag{20}$$

Step 2: Compute the cluster prototypes (means) V_i in (21).

$$V_i = \frac{\sum_{j=1}^{c} U_{ij}^m x^j}{\sum_{j=1}^{c} U_{ij}^m} \quad i = 1, \ldots, c \tag{21}$$

Step 3: Compute the distances in (22).

$$D_{ij}^2 = \left\|x_j - V_i\right\|^2 = (x_j - V_i)(x_j - V_i)^{\mathrm{T}} \tag{22}$$

Step 4: Compute objective function J (23). If $J > \varepsilon_{\text{CLU}}$, go to next step, otherwise stop.

$$J = \sum_{i=1}^{c}\sum_{j=1}^{L}(U_{ij})^m D_{ij}^2 \tag{23}$$

Step 5: Update the partition matrix in (24).

$$U_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{D_{ij}}{D_{kj}}\right)^{2/(m-1)}} \tag{24}$$

3.4.1 Partition validation for the FCM

Partition validation is the problem of finding the best value for c subject to minimization of jth cluster. Cluster validity indices are extremely important for automatically determining the number of clusters.

The partition entropy proposed by [34] is a simple and commonly used cluster validity criteria associated with the FCM algorithm, which is defined as condition (25).

$$H(U,c) = -\frac{1}{L}\sum_{k=1}^{L}\sum_{i=1}^{c} u_{ik}\log(u_{ik}) \tag{25}$$

That is, the optimal clustering means minimizing $H(U, c)$ over the whole c space. However, the computing error of logarithmic function is larger when the value of u_{ik} is very small, which affect the reliability of validation. To overcome this shortcoming, a simpler but more efficient criterion called modified partition entropy [35] is used as condition (26).

$$H_m(U,c) = \frac{1}{L}\sum_{k=1}^{L}\max_{i}(u_{ik})\log(\max_{i}(u_{ik})) \quad i = 1,\ldots,c \tag{26}$$

The partition validation algorithm that uses criteria of (26) is given below:

Step 1: Choose the maximum cluster number $c_{\max}$, iteration limit T, weighting exponent m, and termination criterion $\varepsilon_{\mathrm{CLU}} > 0$

Step 2: Take $c = 2, 3, \ldots, c_{\max}$; initialize the position of cluster centers: $V_0 = (v_{10}, v_{20}, \ldots, v_{c0})$;

Step 3: Take $t = 1, 2, \ldots, T$; calculate u by (24), (22); and calculate V by (21). If $\|V_t - V_{t-1}\| < \varepsilon$, go to next step, otherwise repeat Step 3.

Step 4: Calculate $H_m(c)$ by (26), if $H_m(c-1) < H_m(c)$ and $H_m(c-1) < H_m(c-2)$, then stop, set the optimal cluster number $c = c - 1$; else repeat from Step 2.

The centers of validated clusters are given us $v_j = (v_{j1}, v_{j2}, \ldots, v_{jn}, v_{j,n+1})$; $j = 1, 2, \ldots, c$ and the sigma $(\sigma_j = \sigma_{ji})$ of jth clusters can be roughly calculated [35] as

$$\sigma_i = \left(\sum_{j=1}^{c}\|v_i - v_j\|/c\right)^{0.5} \quad i = 1,\ldots,c \tag{27}$$

3.5 Gustafson–Kessel (GK) algorithm

Gustafson–Kessel extended the standard fuzzy C-means algorithm by employing an adaptive distance norm, in order to detect clusters of different geometrical shapes in one data set. Each cluster has its own norm-inducing matrix (A_i), which yields the inner product norm.

The norm-inducing matrices are $A = (A_1, A_2, \ldots, A_c)$. The objective functional of the GK algorithm is defined by (18). For a fixed A, conditions (21), (24) can be directly applied. However, the objective function (18) cannot be directly minimized with respect to A_i, since its linear in A_i. J (objective function) could be made as small as desired by making A_i less positive definite. To obtain a feasible solution, A_i must be constrained in some way. The usual way of accomplishing this is to constrain the determinant of A_i.

$$|A| = \rho_i, \rho_i > 0, \forall_i \tag{28}$$

Using the Lagrange multiplier method, A_i is obtained for n dimension data.

$$A_i = [\rho_i|F_i|]^{1/n}F^{-1} \tag{29}$$

where F_i is the fuzzy covariance matrix of the ith cluster defined by:

$$F_i = \frac{\sum_{k=1}^{L}(u_{ik})^m(x_k - V_i)(x_k - V_i)^{\mathrm{T}}}{\sum_{k=1}^{L}(u_{ik})^m} \tag{30}$$

Equations (19), (29), (30) give a generalized squared Mahalanobis distance norm between input data (x_k) and cluster mean (V_i) where the covariance is weighted by the membership degrees in U. Without any prior knowledge, the cluster volumes ρ_i are simply fixed at 1 for each cluster. A drawback of the GK algorithm is that due to the constraint (28), it only can find cluster of approximately equal volumes. The eigenstructure of the cluster covariance matrix provides information about the shape and orientation of cluster. The ratio of the lengths of the cluster's hyper-ellipsoid axes is given by ratio of square roots of the eigenvalues of F_i. The directions of the axes are given by the eigenvectors of F_i. An advantage of the GK algorithm over the FCM is that the GK can detect cluster of different shape and orientation in one data set. It is, however, computationally more involved than the FCM, since the inverse and determinant of cluster covariance matrix must be calculated in each iteration [33].

3.5.1 Partition validation for the GK

A recently proposed validity index (V_{SC}) based on the property of membership degree (separation) and the geometric structure of the data set (compactness) is used along

with GK algorithm. The partition validation algorithm adopted here as follows:

$$V_{\mathrm{SC}}(U,V,X) = \frac{\mathrm{Sep}(c)}{\mathrm{Comp}(c)} \tag{31}$$

In this formula, Sep(c) is the fuzzy separation of fuzzy clusters given by

$$\mathrm{Sep}(c) = \mathrm{trace}(S_B) \tag{32}$$

Where S_B is the between-cluster fuzzy scatter matrix, defined as

$$S_B = \sum_{i=1}^{c}\sum_{k=1}^{L}(u_{ki})^m(V_i - \overline{V})(V_i - \overline{V})^{\mathrm{T}} \tag{33}$$

And trace is the sum of the diagonal elements, $\overline{V}$ is the mean values of centers. A large value of Sep(c) indicates that the fuzzy c-partition is characterized by well-separated fuzzy clusters. Comp(c), the total compactness of the fuzzy c-partition, is given by

$$\mathrm{Comp}(c) = \sum_{i=1}^{c}\mathrm{trace}\left(\sum i\right) \tag{34}$$

where $\sum i$ is the fuzzy covariance matrix, defined as

$$\sum i = \frac{\sum_{k=1}^{L}(u_{ik})^m(x_k - V_i)(x_k - V_i)^{\mathrm{T}}}{\sum_{k=1}^{L}(u_{ik})^m} \tag{35}$$

In the above definition, the individual compactness of each cluster is measured by the trace of its covariance matrix (34). The numerator on the right side of (35) is the conventional compactness matrix.

3.6 Examples

Two examples are provided to demonstrate basic capabilities of OLS and CPN algorithms. The first example is a test function defined as below and shown in Fig. 4a:

$$x = -2\pi : \pi/10 : 2\pi, \quad y = 2\sin(x/2) + 4\sin(2x) + 3\cos(x)$$

The second example is shown in Fig. 4b that represents three visually distinct clusters. It can be seen from the Fig. 4a, b that OLS finds cluster centers among existing data points, but CPN does usually find centers where no data exist.

4 Experimental results

To evaluate proposed methods in terms of RMS errors, three different sets of data are used as a test-bed and RMS errors are obtained by (36).

$$E_{\mathrm{RMS}} = \sqrt{\frac{1}{L}\sum_{n=1}^{L}(d(n) - y(n))^2} \tag{36}$$

4.1 Example 1: Nonlinear plant modeling problem

$$y(k) = g(y(k-1),(y(k-2)) + u(k) \tag{37}$$

$$g(y(k-1),(y(k-2)) = \frac{y(k-1)y(k-2)(y(k-1)-0.5)}{1 + y^2(k-1) + y^2(k-2)} \tag{38}$$

Modeling the second-order plant given in (37), (38) has been studied in [36–38].The output of plant depends on past outputs [y (k − 1), y (k − 2)] and current input u (k). The goal is to approximate the nonlinear component g [y (k − 1), y (k − 2)], which is usually called the unforced system in control literature. 404 simulated data points were generated from the plant model. With the starting equilibrium state (0, 0), 202 samples of training data were obtained using a random input signal u (k) that is uniformly distributed in [2.5, 2.5]. The rest of the 202 samples of validation data were obtained using a sinusoidal

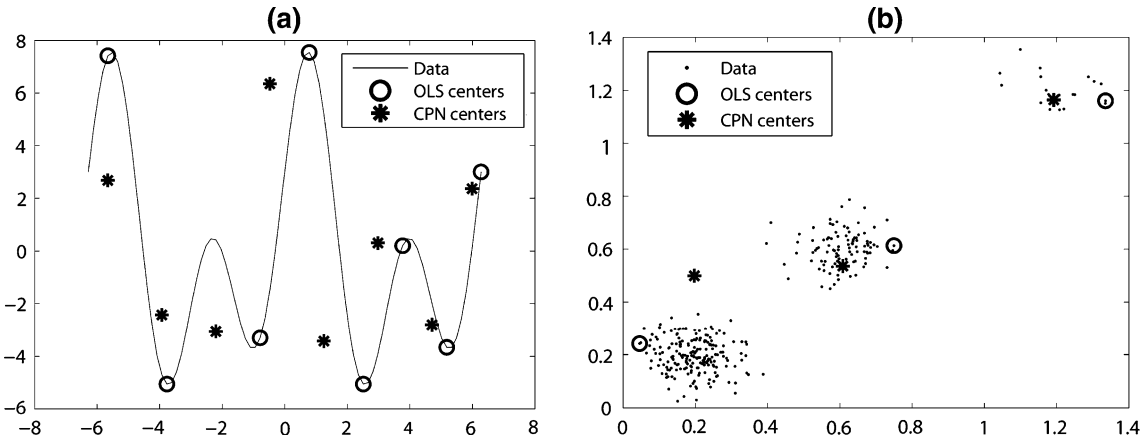

Fig. 4 Centers found by OLS and CPN algorithms

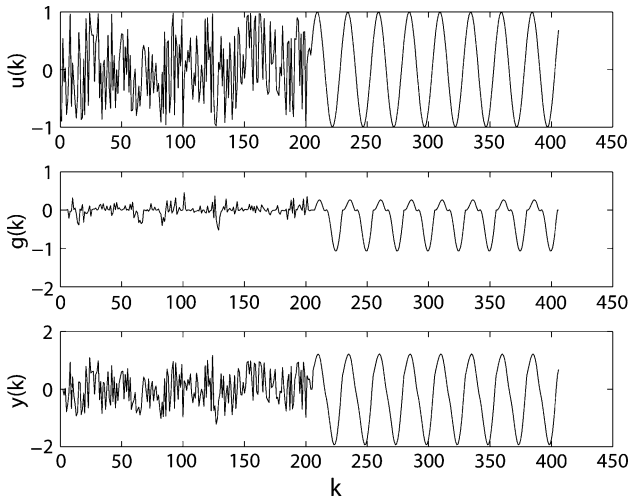

Fig. 5 Input $u(k)$, unforced system $g(k)$ and output $y(k)$ of the plant in (37)

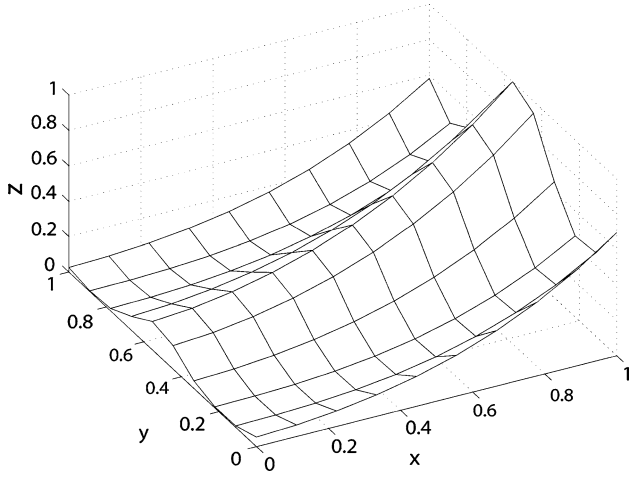

Fig. 6 Nonlinear MISO system

input signal $u\ (k) = \sin(2\pi k/25)$. All data points are normalized in the range of $[-1, 1]$ (Fig. 5).

4.2 Example 2: Nonlinear multiple-input single-output (MISO) system

Nonlinear multiple-input single-output (MISO) System [35]; the problem is two inputs and one output. Hence, each basis function (or cluster) is inherently represented by two input variables, and basis functions can be shown in three dimensions. 100 simulated data points were generated from the plant model using inputs of x; y in [0, 3]. All data points are normalized in the range of [0, 1] and illustrated in Fig. 6.

$$z = (2 + x^{1.5} - 1.5\sin(3y))^2 \tag{39}$$

The proposed hybrid structure can be expressed as IF–THEN rule table that has one dimensional basis functions as shown in Fig. 7a or can be expressed in terms of the equivalent three dimensional basis functions as illustrated in Fig. 7b.

4.3 Example 3: Modeling of chaotic time series

The last application of the proposed methods is to predict complex time series [23, 39–41], a special function approximation problem that arises in such real-world problems as detecting arrhythmia in heartbeats.

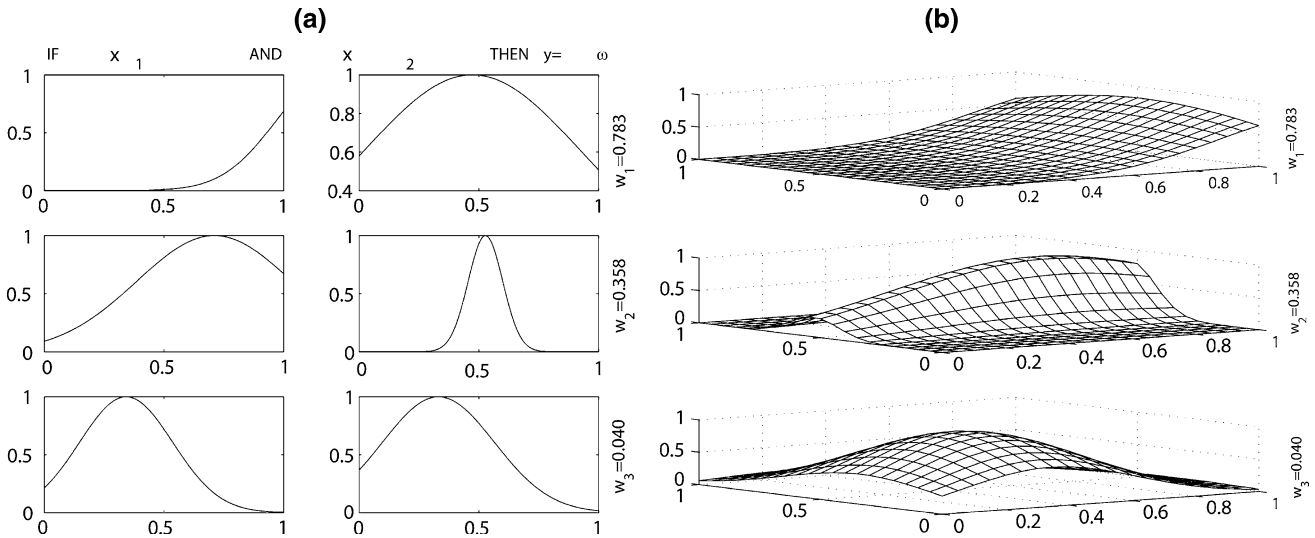

Fig. 7 The IF–THEN representation (**a**). The equivalent basis functions in 3-D (**b**)

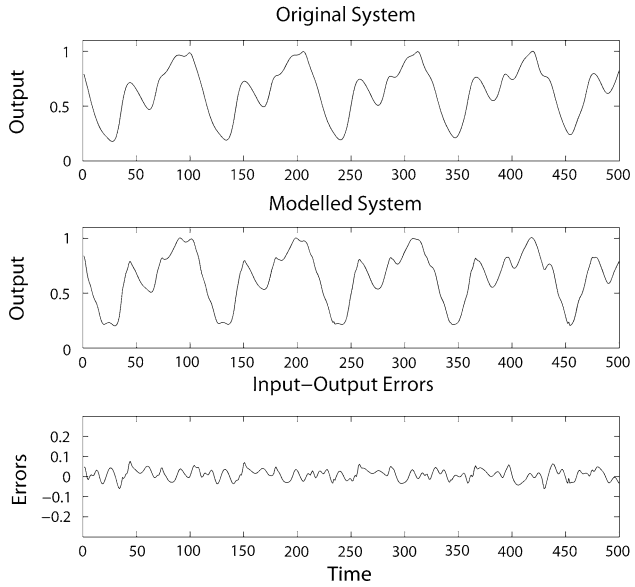

Fig. 8 The chaotic Mackey–Glass time series

The chaotic Mackey–Glass differential is generated from the following delay differential (40) where $\tau = 17$ and the first 500 data ($x(t-3)$, $x(t)$ and $x(t+3)$) is obtained and normalized in the range of [0, 1].

$$\frac{\mathrm{d}x(t)}{\mathrm{d}(t)} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{40}$$

After training completed for all methods, the outcomes of only one selected method (OLS + GK + GD) are graphically presented to save the space. Original system output, modeled system output, and the input–output error between them for Mackey–Glass time series are given in Fig. 8.The proposed methods are compared and the results are given in Table 1. The parameters that are indicated in the table are determined as follows: $\varepsilon_{\mathrm{OLS}}$ and $\varepsilon_{\mathrm{CPN}}$ are determined randomly by user. To ensure to have similar initial conditions, these mentioned parameters are set through the trial and error technique. The μ_c, μ_σ, μ_w learning, and N_{GD} epoch parameters that used widely in artificial neural networks are determined to have the best performance by user.

5 Discussion

It can be seen from Table 1 that RMS errors are significantly high when no pre-processing unit is employed. It is worthy to state that algorithm applied in the fine-tuning step (gradient descent method) of this work is not one of the most efficient algorithms. It is only used here for the

Table 1 The proposed methods are compared with three test functions and results are presented

Examples	Method	OLS + GK + GD	CPN + GK + GD	OLS + FCM + GD	CPN + FCM + GD	Without pre-processing
Nonlinear plant modeling	Initial cluster number	7	11	7	11	6
	Final cluster number	6	6	5	6	
	Time (s)	22.55	18.51	18.42	21.16	17.01
	RMS error (training)	0.0139	0.0144	0.0274	0.0190	0.1601
	RMS error (test)	0.25	0.2493	0.2480	0.2513	0.4338
	Parameters	$\varepsilon_{\mathrm{OLS}} = 0.068$, $\varepsilon_{\mathrm{CPN}} = 0.5$, $\mu_{\mathrm{C}} = \mu_\sigma = \mu_\omega = 0.4$, $N_{\mathrm{GD}} = 200$				
Nonlinear MISO system	Initial cluster number	5	3	5	3	3
	Final cluster number	3	3	3	3	
	Time (s)	4.42	3.81	4.65	3.93	3.72
	RMS error	0.0520	0.0590	0.0593	0.0593	0.1075
	Parameters	$\varepsilon_{\mathrm{OLS}} = 0.09$, $\varepsilon_{\mathrm{CPN}} = 0.75$, $\mu_{\mathrm{C}} = \mu_\sigma = \mu_\omega = 0.4$, $N_{\mathrm{GD}} = 100$				
Mackey–Glass time series	Initial cluster number	12	10	12	10	9
	Final cluster number	7	9	5	5	
	Time (s)	52.52	28.63	56.46	23.43	15.82
	RMS error	0.0174	0.0300	0.0191	0.0191	0.0445
	Parameters	$\varepsilon_{\mathrm{OLS}} = 0.001$, $\varepsilon_{\mathrm{CPN}} = 0.13$, $\mu_{\mathrm{C}} = \mu_\sigma = \mu_\omega = 0.05$, $N_{\mathrm{GD}} = 50$				

sake of clarity and simplicity. Therefore, the initial RBF model can then be optimized by existing algorithms in literature (some hybrid algorithm can be used in fine-tuning step) in a great extend to construct the final model in the future research.

6 Conclusions

Initially, some well-known algorithms (OLS, CPN, FCM, and GK) are reviewed in a detailed manner. Then, a systematic construction of two-step pre-processing units to initialize the Gaussian radial basis function (RBF) neural network is presented in a unifying fashion. The first step, OLS or CPN algorithms only generate M-node initial RBF model from I/O data set, locations of nodes/clusters and associating weights in output layer may not be optimal. Therefore, the second step is dealt with the final structure identification and rough-tuning of associated parameters.

In brief, four pre-processing units are formed and simulated. The proposed pre-processing units are also compared with three non-linear input–output data sets in terms of RMS errors. All comparisons are conducted with as much as similar (or if possible with an identical) number of sub-clusters for each test case. Hence, the effect of structural and parametric initializations of RBF network is demonstrated without bias. Thus, comparative research found that pre-processing unit-1 has the lowest RMS errors for all test cases with a little margin. The proposed pre-processing units can be used to construct an initial RBF (or computationally similar structure such as TS type fuzzy system) model to evaluate and further optimization. The main benefit of the pre-processing units is to construct RBF network with a few basis functions, reducing complexity of network in terms of computational resources and readability.

Acknowledgments This work has been partly supported by Ondokuz Mayıs University Research Foundation Grant (PYO.MUH. 1906.10.001-BAL-LAB).

References

1. Uykan Z, Guzelis C, Celebi M, Koivo H (2000) Analysis of input-output clustering for determining centers of rbfn. IEEE Trans Neural Netw 11(4):851–858
2. Chen S, Cowan CFN, Grant PM (1991) Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Netw 2:302–309
3. Sherstinsky A, Picard RW (1996) On the efficiency of the orthogonal least squares training method for radial basis function networks. IEEE Trans Neural Netw 7:1995–2000
4. Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1(2):281–294
5. Gonzales J, Rojas I, Ortega J (2003) Multiobjective evolutionary optimization of the size, shape and position parameters of radial basis function networks for function approximation. IEEE Trans Neural Netw 14(6):1478–1495
6. Buchtala O, Klimek M, Sick B (2005) Evolutionary optimization of radial basis function classifiers for data mining applications. IEEE Trans Syst Man Cybern Part B Cybern 35:928–947
7. Seng TL, Khalid M, Yusof R (1999) Tuning of a neuro-fuzzy controller by genetic algorithm. IEEE Trans Syst Man Cybern B Cybern 29:226–236
8. Zhou P, Li D, Wu H, Cheng F (2011) The automatic model selection and variable kernel width for RBF neural networks. Neurocomputing 74(17):3628–3637
9. Du D, Li K, Fei M (2010) A fast multi-output RBF neural network construction method. Neurocomputing 73(10–12):2196–2202
10. Lin G-F, Wu M-C (2011) An RBF network with a two-step learning algorithm for developing a reservoir inflow forecasting model. J Hydrol 405:439–450
11. Niros AD, Tsekouras GE (2012) A novel training algorithm for RBF neural network using a hybrid fuzzy clustering approach. Fuzzy Sets Syst 193:62–84
12. Montazer GA, Sabzevari R, Ghorbani F (2009) Three-phase strategy for the OSD learning method in RBF neural networks. Neurocomputing 72(7–9):1797–1802
13. Pedrycz W (1998) Conditional fuzzy clustering in the design of radial basis function neural networks. IEEE Trans Neural Netw 9:601–612
14. Gonzales J, Rojas I, Pomares H, Ortega J, Prieto A (2002) A new clustering technique for function approximation. IEEE Trans Neural Netw 13(1):132–142
15. Staiano A, Tagliaferri R, Pedrycz W (2006) Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering. Neurocomputing 69:1570–1581
16. Billings SA, Wei H-L, Balikhin MA (2007) Generalized multiscale radial basis function networks. Neural Netw 20:1081–1094
17. Roman Neruda PK (2005) Learning methods for radial basis function networks. Future Gener Comput Syst 21:1131–1142
18. Peng J-X, Kang Li, Huang D-S (2006) A hybrid forward algorithm for RBF neural network construction. IEEE Trans Neural Netw 17:1439–1451
19. Carvalho AD, Brizzotti MM (2001) Combining RBF networks trained by different clustering techniques. Neural Process Lett 14:227–240
20. McLoone S, Brown M, Irwin G, Lightbody A (1998) A hybrid linear/nonlinear training algorithm for feedforward neural networks. IEEE Trans Neural Netw 9(4):669–684
21. Kayhan G, Özdemir AE, Eminoglu I (2009) Designing pre-processing units for rbf networks part-1: inital structure identification. INISTA (international symposium on innovations in intelligent systems and applications), Trabzon, pp 185–189
22. Özdemir AE, Kayhan G, Eminoglu I (2009) Designing pre-processing units for RBF networks part-2: final structure identification and coarse tuning of parameters. INISTA (international symposium on innovations in intelligent systems and applications), Trabzon, pp 190–194
23. Chen M-Y, Linkens D (2001) A systematic neuro-fuzzy modeling framework with application to material property prediction. IEEE Trans Syst Man Cybern Part B Cybern 31:781–790
24. Linkens D, Chen M-Y (1999) Input selection and partition validation for fuzzy modeling using neural network. Fuzzy Sets Syst 107:299–308
25. Jang J-SR, Sun C-T (1993) Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Trans Neural Netw 4(1):156–159

26. Hunt KJ, Haas R, Murray-Smith R (1996) Extending the functional equivalence of radial basis function networks and fuzzy inference systems. IEEE Trans Neural Netw 7:776–781
27. Anderson HC, Lotfi A, Westphal LC, Jang JR (1998) Comments on "functional equivalence between radial basis function networks and fuzzy inference systems" [and reply]. IEEE Trans Neural Netw 9(6):1529–1532
28. Nie J, Linkens DA (1993) Learning control using fuzzified self-organizing radial basis function network. IEEE-FS 1:280–287
29. Haykin S (1994) Neural networks: a comprehensive foundation. Prentice Hall, Englewood Cliffs
30. Jenison R, Fissell K (1995) A comparison of the von Mises and Gaussian basis functions for approximating spherical acoustic scatter. IEEE Trans Neural Netw 6(5):1284–1287
31. Xia Q, Wang MY (2006) Orthogonal least squares of unity surface reconstruction with radial basis function. Geom Model Imaging New Trends 28–33
32. Nie J, Linkens DA (1994) Fast self-learning multivariable fuzzy controllers constructed from a modified CPN network. Int J Control 60(3):369–393
33. Babuska R (1998) Fuzzy modeling for control. Kluwer, Dordrecht
34. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Kluwer, Norwell
35. Chen MY (1998) Integrated knowledge-based intelligent control systems: principles, methodologies and practice. PhD thesis, The University of Sheffield
36. Setnes M, Roubos H (2000) Ga-fuzzy modeling and classification: complexity and performance. IEEE Trans Fuzzy Syst 8: 509–522
37. Yen J, Wang L (1998) Application of statistical information criteria for optimal fuzzy model construction. IEEE Trans Fuzzy Syst 6:362–372
38. Yen J, Wang L (1999) Simplifying fuzzy rule-based models using orthogonal transformation methods. IEEE Trans Syst Man Cybern Part B Cybern 29:13–24
39. Chng E, Chen S, Mulgrew B (1996) Gradient radial basis function networks for nonlinear and no stationary time series prediction. IEEE Trans Neural Netw 7(1):190–194
40. Duro RJ, Reyes JS (1999) Discrete-time back propagation for training synaptic delay-based artificial neural networks. IEEE Trans Neural Netw 10(4):779–789
41. Jang J-SR (1993) Anfis: adaptive-network-based fuzzy inference system. IEEE Trans Syst 23:665–685