Radial Basis Function Network Training Using a Nonsymmetric Partition of the Input Space and Particle Swarm Optimization

Alex Alexandridis, *Member, IEEE*, Eva Chondrodima, and Haralambos Sarimveis

***Abstract*— This paper presents a novel algorithm for training radial basis function (RBF) networks, in order to produce models with increased accuracy and parsimony. The proposed methodology is based on a nonsymmetric variant of the fuzzy means (FM) algorithm, which has the ability to determine the number and locations of the hidden-node RBF centers, whereas the synaptic weights are calculated using linear regression. Taking advantage of the short computational times required by the FM algorithm, we wrap a particle swarm optimization (PSO) based engine around it, designed to optimize the fuzzy partition. The result is an integrated framework for fully determining all the parameters of an RBF network. The proposed approach is evaluated through its application on 12 real-world and synthetic benchmark datasets and is also compared with other neural network training techniques. The results show that the RBF network models produced by the PSO-based nonsymmetric FM algorithm outperform the models produced by the other techniques, exhibiting higher prediction accuracies in shorter computational times, accompanied by simpler network structures.**

***Index Terms*— Fuzzy means algorithm, fuzzy partition, nonsymmetric partition, particle swarm optimization, radial basis function.**

I. Introduction

RADIAL basis function (RBF) networks constitute a neural network architecture that has been used extensively for modeling and control of nonlinear systems. The popularity of the RBF network architecture stems from the inherent simplicity of its structure, in contrast to the more complicated multilayer perceptron (MLP) networks. RBF networks comprise a single hidden layer, attached linearly to the output unit of the network. The aforementioned characteristics are reflected on the training algorithms used for RBF networks, which are usually comparatively fast and highly efficient.

In RBF networks, learning essentially corresponds to finding the multidimensional surface that best approximates a set of training examples. This surface is put together as a summation of a number of simpler surfaces, with radial basis symmetry around centers located in specific points of the input space. Training of an RBF network is then equivalent to calculating values for the following parameters:

- number of RBFs;
- coordinates of RBF centers;
- widths of RBFs (applicable only when using RBFs with variable width, e.g., the Gaussian function);
- synaptic weights.

A variety of algorithms have been proposed in the literature for determining the RBF center locations. Preliminary studies [1] and [2] selected the number of RBFs to be equal to the total number of training data, placing the center of each RBF on top of each datapoint. However, this technique ends up in selecting an excessive number of hidden nodes, especially in the case of large datasets. In order to avoid this phenomenon, a clustering technique can be employed, aiming to segment the input space into a number of regions, which is smaller compared to the number of training data. RBFs are then placed at the center of each cluster of data. Based on this concept, the k-means clustering algorithm became a popular approach to the selection of RBF centers in some of the early studies on RBF networks [3], [4]. There are, however, two intrinsic disadvantages associated with the use of k-means. The first is due to its iterative nature, which can lead to long convergence times, and the second originates from its inability to automatically determine the number of RBF centers, thus resulting in a time-consuming trial-and-error procedure for establishing the size of the hidden layer. A multitude of alternative techniques have been proposed to tackle these disadvantages. A significant portion of these methodologies use a constructive approach, building the hidden layer incrementally until a criterion is met. Within this context, the application of the orthogonal least squares algorithm has been thoroughly explored [5]–[9], but other methods have also been proposed, including constructive decay [10], resource allocating networks [11], and the minimum description length principle [12]. Taking a different approach compared to the constructive methods, the fuzzy means (FM) algorithm [13] selects the number of RBF centers and their locations in short computational times, based on fuzzy clustering [14]–[16] of the input space. An online version of the FM algorithm has also been proposed [17], allowing for real-time adaptation of the RBF network structure. Recently, we introduced a nonsymmetric approach for partitioning the input space. Preliminary results [18] have shown that the nonsymmetric partition can lead to the development of more accurate RBF models, with a smaller number of hidden layer nodes.

Manuscript received March 26, 2012; revised August 14, 2012; accepted November 6, 2012. Date of publication December 27, 2012; date of current version January 11, 2012.

A. Alexandridis and E. Chondrodima are with the Department of Electronics, Technological Educational Institute of Athens, Aigaleo 12210, Greece (e-mail: alexx@teiath.gr; evachon@teiath.gr).

H. Sarimveis is with the Department of Chemical Engineering, National Technical University of Athens, Zografou 15780, Greece (e-mail: hsarimv@chemeng.ntua.gr).

Digital Object Identifier 10.1109/TNNLS.2012.2227794

As far as the selection of node widths is concerned, the standard approach involves the p-nearest neighbor heuristic [19], where the widths for each RBF are selected so as to overlap with its p nearest RBFs. More elaborate methods have been suggested [20]–[22] for optimizing the RBF widths in order to improve approximation accuracy.

Taking advantage of the linear connection between the hidden and output layer, most training algorithms calculate the synaptic weights of RBF networks by applying linear regression of the output of the hidden units on the target values. Alternative approaches for calculating the weights include gradient descent methods [23], fuzzy logic [24], and the expectation-maximization algorithm [25].

A few algorithms aspiring to determine all the RBF training parameters in one step have also been proposed in the literature. In [26], a hierarchical Bayesian model is introduced for training RBFs. The model treats all the training parameters as unknown random variables and Bayesian calculation is performed through a reversible-jump Markov chain Monte Carlo method, whereas the networks are optimized using a simulated annealing algorithm. In [27], RBF parameters are determined in a one-step algorithm in interpolation problems with equally spaced nodes, after replacing the Euclidean norm associated to Gaussian RBF with a Mahalanobis norm. In [28], all the RBF network parameters, including input weights on the connections between input and hidden layers, are adjusted by a second-order update rule.

It should be noted, however, that calculating optimal values for all the RBF parameters is a rather cumbersome task. When viewing the RBF network training procedure as an optimization problem, one realizes that the objective function usually presents some rather unwelcome properties, including multimodality, nondifferentiability, and high levels of noise. As these characteristics make use of standard optimization methods inefficient, it is no surprise that a significant number of studies have focused on optimizing the RBF training procedure through the use of alternative approaches, such as evolutionary-based computation techniques. The resulting methodologies include a genetic algorithm for optimizing the number and coordinates of RBF centers [29], a hybrid multilogistic methodology applying evolutionary programming for producing RBFs with simpler structures [30], a multiobjective evolutionary algorithm to optimize RBF networks including some new genetic operators in the evolutionary process [31], and an evolutionary algorithm that performs feature and model selection simultaneously for RBF classifiers in reduced computational times [32].

Another methodology drawn from the arsenal of evolutionary computation methods that has been exploited for RBF network training, is the particle swarm optimization (PSO) technique. PSO is a powerful stochastic optimization algorithm that has been used successfully in conjunction with other computational intelligence tools [33], [34]. A PSO-aided orthogonal forward regression algorithm based on leave-one-out criteria is developed in [35] in order to construct parsimonious RBF networks with tunable nodes. A recursive orthogonal least squares algorithm has been combined with PSO in a novel heuristic structure optimization method for RBF probabilistic networks [36]. An advanced architecture of k-means clustering-based polynomial RBF networks has been introduced in [37], designed using PSO and differential evolution.

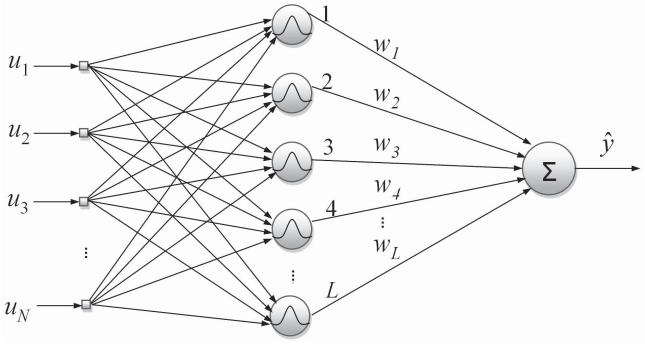

Fig. 1. Typical structure of an RBF network with N input variables, L hidden nodes, and one output variable.

The contribution of this paper is a novel integrated framework for full RBF network training, combining the nonsymmetric variant of the FM algorithm with a specially designed PSO-based optimizer. The nonsymmetric FM (NSFM) algorithm is applied to calculate the number and locations of hidden node centers, while the PSO optimizer is wrapped around it, optimizing the input space fuzzy partition. The combined PSO-NSFM approach results in RBF models with higher accuracy and parsimony, while a satisfactory model can be achieved in relatively small computational times.

The rest of this paper is organized as follows: In the next section, we describe briefly the RBF architecture. Section III presents the FM algorithm, focusing on the nonsymmetric variant. Section IV presents the PSO-NSFM algorithm, starting with a short introduction to PSO and then elaborating on its use for optimizing nonsymmetric fuzzy partitioning, and ultimately for fully training an RBF network. A series of experiments in real and synthetic datasets, evaluating the PSO-NSFM algorithm and comparing it with other methodologies, is presented in Section V. This paper concludes by outlining the advantages of the proposed approach and setting directions for future work.

II. RBF Neural Networks

Fig. 1 depicts the typical structure of an RBF network. The input layer distributes the N input variables to the L nodes of the hidden layer. Each node in the hidden layer is associated with a center, equal in dimension with the number of input variables. Thus, the hidden layer performs a nonlinear transformation and maps the input space onto a new higher dimensional space. The activity $\mu_l(\mathbf{u}(k))$ of the lth node is the Euclidean norm of the difference between the kth input vector and the node center and is given by

$$\mu_l(\mathbf{u}(k)) = \left\|\mathbf{u}(k) - \hat{\mathbf{u}}_l\right\| = \sqrt{\sum_{i=1}^{N}\left(u_i(k) - \hat{u}_{i,l}\right)^2}, \quad k = 1, ..., K \quad (1)$$

where K is the total number of data, $\mathbf{u}^T(k) = [u_1(k), u_2(k), \ldots, u_N(k)]$ is the input vector, and $\hat{\mathbf{u}}_l^T = [\hat{u}_{1,l}, \hat{u}_{2,l}, \ldots, \hat{u}_{N,l}]$ is the center of the lth node.

The activation function for each node is a radially symmetric function. This paper employs the thin plate spline function:

$$g(\mu) = \mu^2 \log(\mu). \tag{2}$$

Thus, the hidden node responses $\mathbf{z}(k)$ become

$$\mathbf{z}(k) = [g(\mu_1(\mathbf{u}(k))), g(\mu_2(\mathbf{u}(k))), \ldots, g(\mu_L(\mathbf{u}(k)))]. \tag{3}$$

The final output $\hat{y}(k)$ of the RBF network is produced by a linear combination of the hidden node responses as

$$\hat{y}(k) = \mathbf{z}(k) \cdot \mathbf{w} = \sum_{l=1}^{L} w_l g(\mu_l(\mathbf{u}(k))) \tag{4}$$

where $\mathbf{w}^T = [w_1, w_2, \ldots, w_L]$ is a vector containing the synaptic weights.

After fixing the RBF centers and nonlinearities in the hidden layer, the synaptic weights are typically calculated using linear regression of the hidden layer outputs to the real measured outputs (target values). The regression problem can be trivially solved using linear least squares in matrix form as

$$\mathbf{w}^T = \mathbf{Y}^T \cdot \mathbf{Z} \cdot \left(\mathbf{Z}^T \cdot \mathbf{Z}\right)^{-1} \tag{5}$$

where $\mathbf{Z} = [\mathbf{z}(1), \mathbf{z}(2), \ldots, \mathbf{z}(K)]^T$ is a matrix containing the hidden layer outputs for all datapoints, and $\mathbf{Y} = [y(1), y(2), \ldots, y(K)]$ is a vector containing all the target values.

III. FM Algorithm

The FM algorithm [13] is a method for selecting the hidden node centers of an RBF network. It has several advantages over the standard approaches like the k-means algorithm [4], including automatic determination of the size of the network, i.e., the number of hidden nodes, and shorter computational times. The FM algorithm has been used successfully in a number of applications including automatic control of industrial processes [38], intelligent control [39], variable selection problems [40], [41], property estimation in materials science [42], [43], etc. A variant of the FM algorithm utilizing a nonsymmetric fuzzy partition of the input space has been recently proposed [18]. What follows is a brief description of the main concepts behind the FM algorithm, emphasizing on the nonsymmetric version. For more details, the interested reader is referred to the original publications.

Consider a system with N normalized input variables u_i, where $i = 1,\ldots,N$. A nonsymmetric fuzzy partition of the input space implies that the domain of each input variable i is partitioned into a different number s_i of one-dimensional (1-D) triangular fuzzy sets. Each fuzzy set can be written as

$$A_{i,j} = \left\{a_{i,j}, \delta\alpha_i\right\}, \qquad i = 1, \ldots, N, \qquad j = 1, \ldots, s_i \tag{6}$$

where $a_{i,j}$ is the center element of fuzzy set $A_{i,j}$, and $\delta\alpha_i$ is half of the respective width. It should be noted that the widths are different for each input direction, depending on the selected number of fuzzy sets s_i.

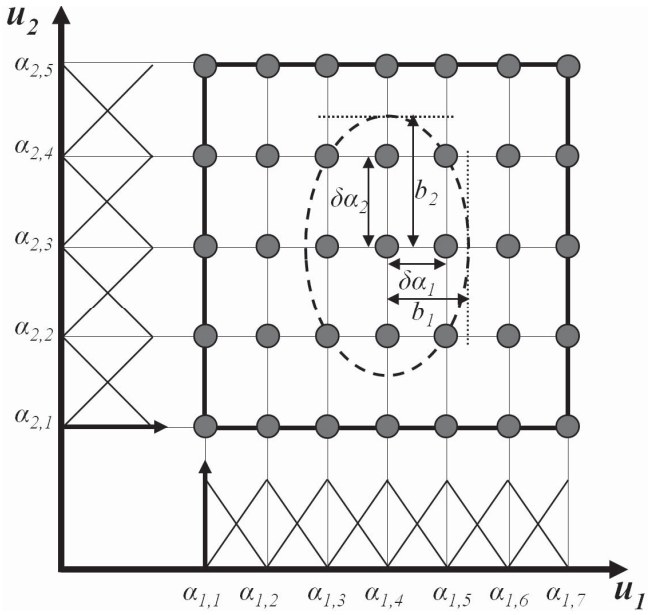

Fig. 2. Nonsymmetric partition on a 2-D input space with an elliptical membership function.

This partitioning technique creates a total of S multidimensional fuzzy subspaces $\mathbf{A}^l$, where $l = 1, \ldots, S$

$$S = \prod_{i=1}^{N} s_i. \tag{7}$$

Each multidimensional fuzzy subspace is generated by combining N 1-D fuzzy sets, one for each input direction. Thus, the fuzzy subspaces can be defined using a center vector $\boldsymbol{\alpha}^l$ and a side vector $\delta\boldsymbol{\alpha}$

$$\mathbf{A}^l = \left\{\boldsymbol{\alpha}^l, \delta\boldsymbol{\alpha}\right\} = \left\{\left[a_{1,j_1}^l, a_{2,j_2}^l, \ldots, a_{N,j_N}^l\right], \left[\delta\alpha_1, \delta\alpha_2, \ldots, \delta\alpha_N\right]\right\}, \qquad l = 1, \ldots, S \tag{8}$$

where a_{i,j_i}^l is the center element of the 1-D fuzzy set A_{i,j_i} that has been assigned to input i. The produced fuzzy subspaces form a grid where each node is a candidate for becoming an RBF center. An example of the nonsymmetric partition is displayed in Fig. 2 using a 2-D case for visualization purposes. The domain of input variable u_1 is partitioned into 7 fuzzy sets, whereas the domain of u_2 is partitioned into 5 fuzzy sets, thus defining a total of 35 fuzzy subspaces. The grid forms 24 equal rectangles in the input space, and the 2 edges of each rectangle are twice the widths of the respective fuzzy sets $\delta\alpha_1$ and $\delta\alpha_2$.

The objective of the FM algorithm is to assemble the RBF network hidden layer by selecting only a small subset of the fuzzy subspaces. The number of the selected fuzzy subspaces should be kept low in order to produce a parsimonious model, but at the same time the produced RBF centers should cover sufficiently the available input data. The FM algorithm performs the selection based on the multidimensional membership function $\mu_{\mathbf{A}^l}(\mathbf{u}(k))$ of an input vector $\mathbf{u}(k)$ to a fuzzy

subspace $\mathbf{A}^l$ [44]

$$\mu_{\mathbf{A}^l}(\mathbf{u}(k)) = \begin{cases} 1 - d_r^l(\mathbf{u}(k)), & \text{if } d_r^l(\mathbf{u}(k)) \le 1 \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where $d_r^l(\mathbf{u}(k))$ is a function of the distance between the fuzzy subspace $\mathbf{A}^l$ and the input data vector $\mathbf{u}(k)$.

The function $d_r^l(\mathbf{u}(k))$ defines a hypersurface on the input space. This hypersurface marks the boundary between input vectors that receive nonzero or zero membership degrees to the fuzzy subspace $\mathbf{A}^l$. In order to account for the nonsymmetric partition, we will associate $d_r^l(\mathbf{u}(k))$ with an N-dimensional hyperellipse of the form

$$\sum_{i=1}^{N} \frac{\left(a_{i,j_i}^l - u_i(k)\right)^2}{(b_i)^2} = 1, \qquad l = 1, \ldots, S \quad (10)$$

where b_i is half of the respective ellipse axis for each variable $i = 1, \ldots, N$. A first prerequisite for calculating b_i is that the surface of the hyperellipse needs to cross all the vertices of the hyper-rectangle defined by the fuzzy subspace; e.g., in the 2-D example of Fig. 2, it should cross the four vertices of the rectangle. However, this condition is not sufficient to uniquely define the hyperellipse, since there are infinite hyperellipses that cross all the vertices. Among them, we select the one whose axes are proportional to the edges of the hyper-rectangle:

$$\frac{\delta\alpha_1}{b_1} = \frac{\delta\alpha_2}{b_2} = \cdots = \frac{\delta\alpha_N}{b_N}. \quad (11)$$

Inclusion of (11) in the formulation of the problem defines uniquely the hyperellipse. It can easily be shown that the hyperellipse is described by the following equation:

$$\sum_{i=1}^{N} \left(\frac{\left(a_{i,j_i}^l - u_i(k)\right)^2}{N(\delta a_i)^2} \right) = 1. \quad (12)$$

Fig. 2 depicts the ellipse that is finally formulated in the 2-D case. Based on (12), the function $d_r^l(\mathbf{u}(k))$ becomes

$$d_r^l(\mathbf{u}(k)) = \sqrt{\sum_{i=1}^{N} \left(\frac{\left(a_{i,j_i}^l - u_i(k)\right)^2}{N(\delta a_i)^2} \right)}. \quad (13)$$

Having defined the membership function, the algorithm proceeds with finding the subset of fuzzy subspaces that assign a nonzero multidimensional degree to all input training vectors. This is accomplished using a noniterative algorithm which requires only one pass of the input data, thus rendering the center calculation procedure extremely fast, even in the presence of a large database of input examples. Algorithm 1 presents an overview of the NSFM algorithm.

As far as the symmetric fuzzy partition employed by the original FM algorithm is concerned, it can emerge as a special case of the nonsymmetric one, assuming that all the input variables are partitioned into the same number of fuzzy sets s.

Algorithm 1 NSFM Algorithm

Input: $\{\mathbf{U}_{\text{train}}, \mathbf{Y}_{\text{train}}\}$: Training Dataset,
$\mathbf{s} = [s_1, s_2, \ldots, s_N]$: Number of fuzzy sets for partitioning each input dimension

Output: L: Number of selected RBF centers,
$\hat{\mathbf{U}} = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_L]^T$: Selected RBF center locations

1: Take the first data point:$k \leftarrow 1$
2: Begin calculations for the first RBF center: $L \leftarrow 1$
3: **For** $i = 1{:}N$ **Do:**
4: Calculate the fuzzy set with maximum membership in each dimension i: $A_i^1 = \{a_i^1, \delta a_i\} \leftarrow \max_{1 \le j \le s_i} \left[\mu_{A_{i,j}}(u_i(1))\right]$
5: **End For**
6: Generate the first RBF center $\hat{\mathbf{u}}_1$:

$$\hat{\mathbf{u}}_1 = [a_1^1, a_2^1, \ldots, a_N^1]$$

7: **For** $k = 2{:}K$ **Do:**
8: **If** data point k lies outside the hyperellipses defined by the already selected centers: $\min_{1 \le l \le L} \left[d_r^l(\mathbf{u}(k))\right] > 1$, where d_r^l is calculated by (13)
9: Add a new RBF center: $L \leftarrow L + 1$
10: **For** $i = 1{:}N$ **Do:**
11: Calculate the fuzzy set with maximum membership in each dimension i: $A_i^L = \{a_i^L, \delta a_i\} \leftarrow \max_{1 \le j \le s_i} \left[\mu_{A_{i,j}}(u_i(k))\right]$
12: **End For**
13: Generate the Lth RBF center $\hat{\mathbf{u}}_L$:

$$\hat{\mathbf{u}}_L = [a_1^L, a_2^L, \ldots, a_N^L]$$

14: **End If**
15: **End for**

This leads to a spherical membership function, through the use of the following distance equation:

$$d_r^l(\mathbf{u}(k)) = \sqrt{\frac{\sum_{i=1}^{N} \left(a_{i,j_i}^l - u_i(k)\right)^2}{\sqrt{N}\delta a}}. \quad (14)$$

In this case, the RBF centers are selected so as to guarantee that all training datapoints are covered by at least one hypersphere. The final selection depends on a single parameter, i.e., the allocated number of fuzzy sets s, which is common for all input variables. This makes it rather easy to optimize the resulting RBF network by testing the few possible fuzzy partitions. In general, using a larger number s creates a denser partition grid and results in the selection of more RBF centers.

Obviously, the original FM algorithm is less flexible and provides fewer design parameters compared to the nonsymmetric one. In a recent publication [18], we have shown that the NSFM algorithm leads to significant improvements in both the accuracy and parsimony of the produced model. On the other hand, the nonsymmetric partition defines a more complicated network design problem, due to the need to optimally determine an increased number of operational parameters. For problems involving a low number of input variables,

the optimal partition can be calculated using an exhaustive search procedure similar to the one used for the original FM algorithm, where all possible partitions are tested [18]. This approach is feasible for small-scale problems, due to the fast training procedure adopted by the FM algorithm. The additional computational burden imposed by medium or large-scale problems, though, makes the use of an exhaustive search procedure prohibitive. In this paper, we introduce a PSO-based approach for calculating the optimal nonsymmetric partition of the input space, thus leading to a new integrated RBF network training procedure.

IV. PSO-NSFM Algorithm

PSO is a simple, yet effective, optimization algorithm based on simulating the social behavior of birds within a flock. The method is based on a population of possible solutions coded as particles, which "fly" through the hyperdimensional search space in search of the optimal solution. Each individual particle changes its position stochastically, trying to repeat its own successful past positions, and at the same time emulate the success of its neighboring individuals.

A. Standard PSO

Consider an N–dimensional search space where each possible solution is coded as the position of one particle in a swarm containing a total of P particles. The position of particle i at time step t is denoted as $\mathbf{x}_i(t)$, where $i = 1, 2, \ldots, P$. At each time step t, the particle positions are updated by adding a velocity vector $\mathbf{v}_i(t)$, using the following equation:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \tag{15}$$

The velocity vector $\mathbf{v}_i(t)$ is calculated as follows:

$$\begin{aligned} v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + \cdots \\ + c_2 r_{2j}(t)\left[\hat{y}_j(t) - x_{ij}(t)\right] \end{aligned} \tag{16}$$

where

$v_{ij}(t)$	velocity of particle $i = 1,\ldots,P$, in dimension $j = 1,\ldots,N$, at time step t;
$x_{ij}(t)$	position of particle $i = 1,\ldots,P$, in dimension $j = 1,\ldots,N$, at time step t;
$y_{ij}(t)$	personal best position of particle $i = 1,\ldots,P$, in dimension $j = 1,\ldots,N$, at time step t;
$\hat{y}_j(t)$	global best position of the entire swarm in dimension $j = 1,\ldots,N$, at time step t;
c_1 and c_2	acceleration constants;
$r_{1j}(t)$ and $r_{2j}(t)$	random values in the range [0, 1], sampled at each time step t from a uniform distribution.

The personal best position $\mathbf{y}_i(t)$ of each particle i at each time step t is calculated as

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t), & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1), & \text{otherwise} \end{cases} \tag{17}$$

where f is the fitness function.

The global best position $\hat{\mathbf{y}}(t)$ at each time step t is calculated as the best among the best personal positions

$$f\left(\hat{\mathbf{y}}(t)\right) = \min\left(f(\mathbf{y}_1(t)), \ldots, f(\mathbf{y}_P(t))\right). \tag{18}$$

Equation (16) comprises three terms. The first term is known as the memory term and it represents the effect that the particle's previous velocity has on its current velocity.

The second term is known as the cognitive term and it expresses the personal experience of each particle. The cognitive term is proportional to the distance of a particle from its own best position. The constant c_1 acts as the gain of the cognitive term and is also known as nostalgia, as it expresses the desire of each particle to regain its best position.

The third term is known as the social term and it expresses the swarm's collective experience. The social term is proportional to the distance of a particle from the swarm's best position so far. The constant c_2 acts as the gain of the social term and is also known as envy, as it expresses the desire of each particle to perform as good as its neighbors.

As the exploration–exploitation tradeoff plays a vital role in the successful solution of an optimization problem, a mechanism known as "velocity clamping" has been introduced in order to control the exploration and exploitation capabilities of the swarm. Velocity clamping bounds the elements of the velocity vector to stay within predefined values

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1), & \text{if } \left\|v_{ij}(t+1)\right\| < V_{\max} \\ \pm V_{\max}, & \text{otherwise.} \end{cases} \tag{19}$$

Various criteria have been proposed as stopping conditions for the PSO algorithm, including the following:

- maximum number of iterations is reached;
- an acceptable solution is found;
- slope of the objective function becomes very small;
- normalized swarm radius becomes close to zero.

B. PSO for Optimizing Fuzzy Partitioning

The PSO concept can be used as a search method for calculating the optimum nonsymmetric partitioning. In this case, the particles in the swarm should represent different ways of partitioning the input space. Applying this formulation, the elements of each particle $\mathbf{s}_i(t)$ at time step t correspond to the number of fuzzy sets in each dimension

$$\mathbf{s}_i(t) = \left[s_1(t),\ s_2(t),\ \ldots,\ s_N(t)\right]^T. \tag{20}$$

Fig. 3 gives a schematic representation for encoding the nonsymmetric partition into particles, using a 3-D input space. The particular encoding requires alterations to (15) and (16), which, for a basic PSO model with fully connected topology, become

$$\mathbf{s}_i(t+1) = \mathbf{s}_i(t) + \mathbf{v}_i(t+1) \tag{21}$$

$$\begin{aligned} v_{ij}(t+1) = \text{round}\Big\{ v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - s_{ij}(t)\right] \\ + c_2 r_{2j}(t)\left[\hat{y}_j(t) - s_{ij}(t)\right]. \end{aligned} \tag{22}$$

In (22), the round operator is used to round the velocities to the nearest integer, since the number of fuzzy sets in

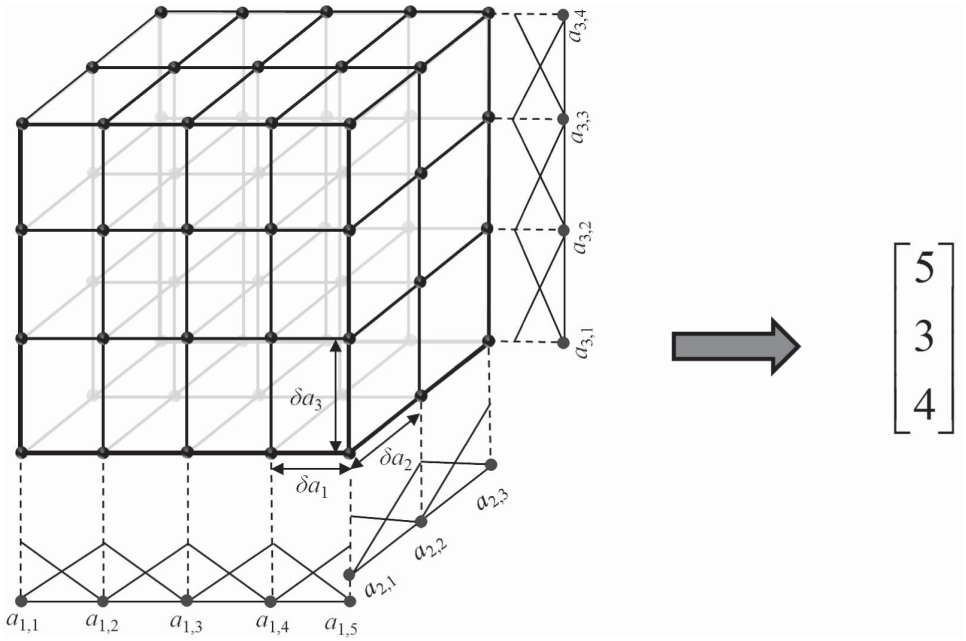

Fig. 3. Encoding a nonsymmetric 3-D input space into a particle: The number of fuzzy sets in each dimension becomes the corresponding element of the particle.

each dimension can receive only integer values. The personal best position of each particle is calculated using (17), after substituting $\mathbf{x}_i(t)$ with $\mathbf{s}_i(t)$, i.e., the particle containing the input partitioning. The global best position and the clamping factor are given by (18) and (19), respectively.

C. PSO-Based RBF Network Training

Optimization of the nonsymmetric fuzzy partitioning of the input space using PSO can form the basis for an integrated RBF network training methodology. The algorithm starts by randomly dividing the available data in three datasets, namely, the training, validation, and testing datasets. The existence of a third independent dataset for testing the produced RBF network is crucial, since the optimization procedure could result in over-fitting the model to the validation dataset. During the particle initialization stage, P different partitions of the input space are selected randomly, and coded as particles. Each particle then gives birth to an RBF network by applying the NSFM algorithm. The latter calculates the number and coordinates of the RBF centers based on (13). After fixing the RBF centers, the synaptic weights can be trivially calculated using (5). Based on the emerging RBF networks, a fitness value is calculated for each particle, by applying an error-related criterion. The proposed algorithm adopts the root mean square error (RMSE) criterion, but alternative error functions can also be used. The resulting fitness values are compared to the personal and global best particle positions, which are conditionally updated according to (17) and (18). The next step is to calculate new particle velocities for time step $t+1$, and then update the particle positions using (21) and (22), respectively. The algorithm then returns to the RBF network generation stage until one of two stopping conditions has been met. The first stopping condition is satisfied when a maximum number of simulation steps is reached, whereas the second is satisfied when the normalized swarm radius becomes smaller than a threshold value. The normalized swarm radius is calculated by

$$R_{\text{norm}}(t) = \frac{R_{\max}(t)}{R_{\max}(1)} \tag{23}$$

where $R_{\max}(t)$ stands for the maximum radius of the swarm at simulation time step t, given by

$$R_{\max}(t) = \max_{1 \le i \le P} \left[\left\| \mathbf{s}_i(t) - \hat{\mathbf{y}}(t) \right\| \right]. \tag{24}$$

When the normalized swarm radius R_{norm} becomes small, the swarm's potential for further improvement diminishes [45], as all the individuals are located in the proximity of the global best position. Algorithm 2 presents an overview of the PSO-NSFM algorithm.

V. Experiments

A. Benchmark Dataset Description

The proposed methodology was evaluated on various synthetic and real-world benchmark datasets. The latter were downloaded from the UCI Machine Learning Repository [46], except from the Gas Furnace dataset, which can be found in [47]. Table I presents a summary of the employed datasets, together with the number of input variables and total data points for each dataset, whereas a brief description for each one of them is given next.

- *Ailerons–Elevators:* These datasets are obtained from flying an F16 aircraft. The attributes describe the status of the airplane, while the goal is to predict the control action on the ailerons and the elevators of the aircraft, respectively.

Algorithm 2 PSO-NSFM Algorithm

Input: $\{\mathbf{U}_{\text{train}}, \mathbf{Y}_{\text{train}}\}$: Training dataset,
$\{\mathbf{U}_{\text{val}}, \mathbf{Y}_{\text{val}}\}$: Validation dataset,
$s_{\min}$, $s_{\max}$: Minimum and maximum number of fuzzy sets,
P: Swarm population,
c_1, c_2, $V_{\max}$: PSO operational parameters affecting the cognitive and social components and velocity clamping, respectively,
ξ_1, ξ_2: parameters for stopping criteria

Output: L_f: Number of selected RBF centers,
$\hat{\mathbf{U}}_f$: Selected RBF center locations,
$\mathbf{w}_f$: Selected RBF synaptic weights

1: Initialize particle coordinates at random integer numbers between $s_{\min}$ and $s_{\max}$: $\mathbf{s}_i(0) \leftarrow \left[\underbrace{\text{rand, rand, } \ldots \text{, rand}}_{N}\right]$
2: Start with the first time step: $t \leftarrow 1$
3: **While** none of the two stopping criteria (ξ_1,ξ_2) has been met **Do:**
4: **For** i=1:P **Do:**
5: Pass $\mathbf{s}_i(t)$ and the training dataset $\{\mathbf{U}_{\text{train}}, \mathbf{Y}_{\text{train}}\}$ to algorithm 1, in order to calculate the total number of RBF centers $L_i(t)$ and their locations $\hat{\mathbf{u}}_i(t)$
6: Use (5) to calculate the synaptic weights $\mathbf{w}_i(t)$
7: Calculate fitness function $f(\mathbf{s}_i(t))$ on the validation dataset $\{\mathbf{U}_{\text{val}}, \mathbf{Y}_{\text{val}}\}$, and update the personal $\mathbf{y}_i(t)$ and global best $\hat{\mathbf{y}}(t)$ positions if needed, using (17) and (18)
8: **End For**
9: **For** i=1:P **Do:**
10: **For** j=1:N **Do:**
11: Update the elements of the velocity vector $v_{ij}(t+1)$ using (22)
12: Perform velocity clamping, if needed, using (19)
13: **End for**
14: Update particle positions $\mathbf{s}_i(t+1)$ using (21)
15: **End For**
16. Proceed to the next time step: $t \leftarrow t+1$
17: **End While**

- *Auto MPG:* A slightly modified version of the original dataset has been employed [48]. The objective is to predict the fuel consumption of an automobile, using as inputs some of its characteristics.
- *Auto Price:* The objective here is to predict the price of an automobile, based on its characteristics. All nominal attributes and data instances with unknowns were removed. This dataset is also known as the Auto Mobile dataset.
- *Boston Housing:* This dataset concerns the prediction of selling prices for houses in Boston, based on characteristics of the houses. Data were acquired from the 1970 census.

TABLE I
BENCHMARK DATASETS

Dataset	No. of Inputs	No. of Examples
Real-World Datasets		
Ailerons	40	13 750
Auto MPG	6	392
Auto Price	15	159
Boston Housing	13	506
CPU small	12	8192
Elevators	18	16 599
Gas Furnace	3	290
House Price-16H	16	22 784
Machine CPU	6	209
Synthetic Datasets		
Friedman	5	1000
Mackey-Glass	6	1995
Samad	3	1500

- *CPU Small:* CPU small is a small variant of the Computer Activity dataset. It comprises a collection of computer-system activity measures from a Sun SPARC-station. The task is to predict the portion of time that the CPUs run in user mode,taking into account a restricted number of attributes.
- *Friedman:* This dataset is generated from the function [49]

$$y = 5\left[2\sin(\pi x_1 x_2) + 4(x_3 - 0.5)^2 + 2x_4 + x_5\right] + \varepsilon \tag{25}$$

where ε is Gaussian noise $\sim N(0, 0.8)$. Values for the input variables are sampled from a uniform distribution in [0,1].
- *Gas Furnace:* This dataset involves prediction of the CO_2 concentration $y(t)$ in the exhaust gases of a gas furnace system, while changing the methane feed rate $u(t)$. A previous study [41] has shown that the optimum combination of input variables results in a model of the form

$$y(t) = \text{RBF}(u(t-2), y(t-2), y(t-1)). \tag{26}$$

- *House Price-16H:* This dataset is concerned with predicting the median price of houses based on demographic composition and the state of the housing market.
- *Machine CPU:* The objective is to predict the relative CPU performance using as inputs CPU characteristics. This dataset is also known as the Computer Hardware dataset.
- *Mackey-Glass:* The Mackey-Glass dataset originates from a chaotic time series which has been used extensively for evaluating neural network models [50]–[52]; for the discrete case, it is generated from the following difference equation:

$$y(t+1) = (1-b)\,y(t) + a\frac{y(t-\tau)}{\left(1 + y^{10}(t-\tau)\right)}. \tag{27}$$

The objective is to build a model for predicting the current value of the time series $y(t)$, based on the previous six values

$$y(t) = \text{RBF}\left(y(t-1), y(t-2), \ldots, y(t-6)\right). \tag{28}$$

The parameter values $a = 0.2$, $b = 0.1$, and $\tau = 30$ have been used in (27) for generating the dataset. It should be noted that the first 1000 values were discarded, in order to allow the system to reach steady-state operation. Gaussian noise $\sim N(0, 0.05)$ was added to the output $y(t)$.

- *Samad:* This dataset is generated from the function [53]

$$y = \frac{1}{1 + \exp\left[-\exp(x_1) + (x_2 - 0.5)^2 + \sin(\pi x_3)\right]} + \varepsilon \tag{29}$$

where ε is Gaussian noise $\sim N(0,0.025)$. Values for the input variables are sampled from a uniform distribution in [0,1].

B. Results and Discussion

For each case, data were split randomly to training, validation, and testing datasets. In general, there is no consensus in the literature regarding the amount of data that should be allocated to each dataset. In this paper, a 50% - 25% - 25% ratio (training - validation - testing) was adopted, and data splitting was performed in a random number. The only exceptions to the random splitting rule were the Gas Furnace and Mackey-Glass datasets, where common practice is to maintain the original data order.

Table II presents the operational parameters for the PSO-NSFM algorithm (including the parameters used by the NSFM algorithm and the PSO optimizer) applied to all experiments. The minimum and maximum number of fuzzy sets define the lower and upper bounds of the search space, respectively. Previous experiments with both symmetric and nonsymmetric versions of the FM algorithm have shown that partitions outside these bounds are not likely to result to successful models. The constants c_1 and c_2 controlling the effect of the cognitive and social components, respectively, were found to have only a small effect on the outcome of the optimization procedure when kept at small values. Selection of larger values for c_1 and c_2, however, resulted in poor exploitation capabilities of the algorithm. Furthermore, equal values were selected for c_1 and c_2, since previous studies have shown that PSO is usually most effective when the two constants coexist in good balance [45]. The exploitation–exploration tradeoff was controlled by the velocity clamping constant $V_{\max}$. As the size of the search space increases with the dimensionality of the input space, particle velocities should be allowed to increase in order to guarantee that the search space will be sufficiently explored within reasonable simulation times. Therefore, three different velocity clamping constants were used, depending on the input space dimensionality of the problem (small problems: 1–4 input variables, medium problems: 5–8 input variables, large problems: more than 8 input variables).

TABLE II
OPERATIONAL PARAMETERS FOR THE PSO-NSFM ALGORITHM

Parameter	Symbol	Value
Minimum number of fuzzy sets	$s_{\min}$	4
Maximum number of fuzzy sets	$s_{\max}$	50
Population	P	20
Nostalgia	c_1	0.05
Envy	c_2	0.05
Velocity clamping constant[a]	V_{max}	S:5, M:15, L:25
Maximum number of simulation steps	ξ_1	8000
Minimum normalized swarm radius	ξ_2	0.1

[a] Value depending on input space dimensionality: small problems (S): 1–4 input variables, medium problems (M): 5–8 input variables, large problems (L): more than 8 input variables.

For comparison purposes, two additional methodologies were also tested, namely, RBF networks trained with the SFM algorithm and MLP networks trained with the Levenberg-Marquardt algorithm [54]. In both cases, the network parameters were determined using the training dataset, and then model selection was performed using the validation dataset. In the case of RBF networks trained with SFM, model selection was controlled by only one parameter, namely, the number of fuzzy sets s for partitioning each input variable. Selection of s was made after testing all partitions from 4 to 50 fuzzy sets. Regarding MLP networks, a standard two-hidden-layer architecture was postulated and then an exhaustive search was performed, testing all possible combinations when the number of nodes in each hidden layer ranged from 5 to 40.

The results for all the datasets are summarized in Tables III and IV. Table III depicts the RMSE in the validation and testing datasets and the number of nodes for all three methodologies, together with the selected fuzzy partition for the two RBF models. The computational time for all three methodologies and the simulation time steps (STSs) for PSO-NSFM algorithm completion are shown in Table IV, together with the computational time and STSs needed for the PSO-NSFM algorithm to generate a model that outperforms the SFM algorithm in terms of the RMSE fitness criterion. All computational times were measured on a PC with Intel Core 2 Quad processor (2.83 GHz) with 4 GB of RAM. Due to the stochastic nature of the PSO method, the PSO-NSFM algorithm returns different results for each run. In order to get consistent results, the algorithm was tested 30 times for each dataset. The tables depict the best result, followed by the average and standard deviation values from the 30 runs in parentheses. Fig. 4 presents the evolution of the RMSE in the validation dataset per STS for the best solution in each case.

It can be seen that in all cases the PSO-NSFM algorithm outperforms the SFM algorithm and the MLP networks in terms of the best solution found in the validation and testing datasets. As far as the average RMSE value found by PSO-NSFM is concerned, it is lower compared to the ones found by the other two methodologies in 10 out of 12 cases, considering either the validation or the testing dataset. Standard

TABLE III
RESULTS FOR ALL DATASETS–RMSE, FUZZY PARTITION AND NUMBER OF NODES

Dataset	Algorithm	RMSE Validation	RMSE Testing	Fuzzy Partition	Number of Nodes[a]
Ailerons	PSO-NSFM	1.57×10^{-4} ($1.58\times10^{-4}\pm1.08\times10^{-6}$)	1.56×10^{-4} ($1.56\times10^{-4}\pm1.70\times10^{-6}$)	[4 6 4 4 13 7 5 4 14 11 22 32 5 4 4 10 5 37 12 9 4 50 4 24 50 37 48 44 5 50 4 27 6 50 42 6 4 50 9 9]	237 (314±127)
	SFM	1.59×10^{-4}	1.57×10^{-4}	13	311
	MLP	1.59×10^{-4}	1.58×10^{-4}	-	[35 25]
Auto MPG	PSO-NSFM	2.11 ($2.14\pm3.19\times10^{-2}$)	1.95 ($2.09\pm8.96\times10^{-2}$)	[27 4 4 32 4 8]	36 (39±24)
	SFM	2.22	2.12	10	41
	MLP	2.51	2.34	-	[23 19]
Auto Price	PSO-NSFM	2.17×10^{3} ($2.38\times10^{3}\pm1.28\times10^{2}$)	2.11×10^{3} ($2.30\times10^{3}\pm9.06\times10^{1}$)	[4 4 29 50 50 23 7 34 5 10 24 4 35 4 37]	57 (57±5)
	SFM	2.73×10^{3}	2.86×10^{3}	11	44
	MLP	2.29×10^{3}	2.38×10^{3}	-	[17 11]
Boston Housing	PSO-NSFM	2.35 ($2.38\pm3.13\times10^{-2}$)	2.72 ($3.12\pm2.26\times10^{-1}$)	[48 50 39 4 50 24 4 6 28 45 18 5 11]	138(141±17)
	SFM	2.77	2.93	16	146
	MLP	2.47	3.38	-	[18 19]
CPU Small	PSO-NSFM	2.80 ($2.83\pm1.34\times10^{-2}$)	2.81 ($2.83\pm2.39\times10^{-2}$)	[4 7 18 15 16 32 24 29 26 16 35 20]	467(463±261)
	SFM	2.87	2.84	23	480
	MLP	3.01	3.02	-	[24 30]
Elevators	PSO-NSFM	2.08×10^{-3} ($2.08\times10^{-3}\pm7.27\times10^{-6}$)	2.11×10^{-3} ($2.11\times10^{-3}\pm1.08\times10^{-5}$)	[16 17 5 18 49 17 13 7 9 16 15 11 4 11 5 15 16 12]	981 (805±274)
	SFM	2.10×10^{-3}	2.12×10^{-3}	18	1776
	MLP	2.10×10^{-3}	2.14×10^{-3}	-	[40 24]
Friedman	PSO-NSFM	9.08×10^{-1} ($9.33\times10^{-1}\pm1.61\times10^{-2}$)	1.06 ($1.07\pm3.03\times10^{-2}$)	[8 4 4 32 4]	114(116±66)
	SFM	9.64×10^{-1}	1.09	8	179
	MLP	1.04	1.08	-	[15 15]
Gas Furnace	PSO-NSFM	1.70×10^{-1} ($1.80\times10^{-1}\pm3.70\times10^{-3}$)	4.27×10^{-1} ($4.04\times10^{-1}\pm1.65\times10^{-2}$)	[8 5 23]	22 (17±5)
	SFM	2.19×10^{-1}	4.44×10^{-1}	15	29
	MLP	3.10×10^{-1}	4.60×10^{-1}	-	[8 19]
House Price-16H	PSO-NSFM	3.62×10^{4} ($3.63\times10^{4}\pm1.24\times10^{2}$)	3.61×10^{4} ($3.59\times10^{4}\pm4.98\times10^{2}$)	[5 9 4 26 39 10 6 4 11 10 24 16 13 12 14 30]	984 (902±299)
	SFM	3.64×10^{4}	3.62×10^{4}	17	1065
	MLP	3.64×10^{4}	3.63×10^{4}	-	[37 14]
Machine CPU	PSO-NSFM	2.28×10^{1} ($2.30\times10^{1}\pm4.03\times10^{-1}$)	2.16×10^{1} ($2.63\times10^{1}\pm4.50$)	[12 4 13 6 16 5]	19 (21±6)
	SFM	2.49×10^{1}	2.77×10^{1}	11	22
	MLP	2.92×10^{1}	2.99×10^{1}	-	[21 7]
Mackey-Glass	PSO-NSFM	4.79×10^{-2} ($4.81\times10^{-2}\pm1.48\times10^{-4}$)	5.30×10^{-2} ($5.31\times10^{-2}\pm2.29\times10^{-4}$)	[4 4 4 15 5 43]	24 (25±4)
	SFM	4.92×10^{-2}	5.34×10^{-2}	17	34
	MLP	4.88×10^{-2}	5.35×10^{-2}	-	[9 13]
Samad	PSO-NSFM	2.47×10^{-2} ($2.48\times10^{-2}\pm2.11\times10^{-4}$)	2.80×10^{-2} ($2.79\times10^{-2}\pm4.40\times10^{-4}$)	[4 6 38]	89 (87±19)
	SFM	2.59×10^{-2}	2.83×10^{-2}	11	112
	MLP	2.49×10^{-2}	2.81×10^{-2}	-	[13 7]

The PSO-NSFM algorithm was tested 30 times for each dataset. The table depicts the run corresponding to the best result in terms of RMSE in the validation dataset, followed by the average and standard deviation values from the 30 runs in parentheses.

[a] The number of nodes for the MLP networks is given in the form: [first layer nodes second layer nodes].

deviation values of RMSE were found to be relatively small, attesting to the fact that the algorithm provides consistent results.

Another important improvement over the SFM algorithm concerns the size of the produced networks. In 10 out of 12 datasets, the PSO-NFSM approach managed to reduce the

TABLE IV
RESULTS FOR ALL DATASETS-COMPUTATIONAL TIME AND SIMULATION TIME-STEPS

Dataset	Computational Time (*s*)				Simulation Time-Steps	
	PSO-NSFM (Complete)	PSO-NSFM (Outperform)	SFM	MLP	PSO-NSFM (Complete)	PSO-NSFM (Outperform)
Ailerons	4220 (7138±3176)	941 (1525±1003)	2613	8141	118 (227±167)	21 (32±21)
Auto MPG	92 (388±113)	4 (6±4)	8	1174	319(1725±584)	7 (13±9)
Auto Price	339 (357±112)	0.7 (5±2)	6	624	832(825±295)	1 (9±4)
Boston Housing	2588 (2824±877)	0.8 (7±3)	13	2402	5709 (6677±2118)	1 (7±2)
CPU Small	4505 (2377±1053)	17 (705±289)	790	2814	547 (261±121)	1 (58±25)
Elevators	7197 (8257±2877)	1745 (2587±1201)	3873	10 715	166 (194±77)	34 (47±20)
Friedman	133 (436±123)	23 (41±23)	47	1300	245 (949±285)	21 (37±19)
Gas Furnace	77 (35±22)	0.4 (1.5±1)	1.6	1040	620 (250±112)	1 (7±5)
House Price-16H	13 201 (11 708±3651)	869 (2109±1108)	3508	15 745	284 (249±86)	18 (44±21)
Machine CPU	83 (69±32)	1 (3±2)	4	954	943 (698±274)	5 (18±13)
Mackey-Glass	539 (303±218)	1 (7±4)	27	1140	3259 (2124±1807)	3 (11±5)
Samad	60 (104±30)	2 (14±11)	34	1298	89(178±72)	1 (10±8)

The PSO-NSFM algorithm was tested 30 times for each dataset. The table depicts the best result, followed by the average and standard deviation values from the 30 runs in parentheses. For the PSO-NSFM (Complete) case, the best result corresponds to the run achieving the lowest RMSE in the validation dataset upon completion of the algorithm, while for the PSO-NSFM (Outperform) case, the best result corresponds to the run achieving the lowest computational time for outperforming the SFM algorithm.

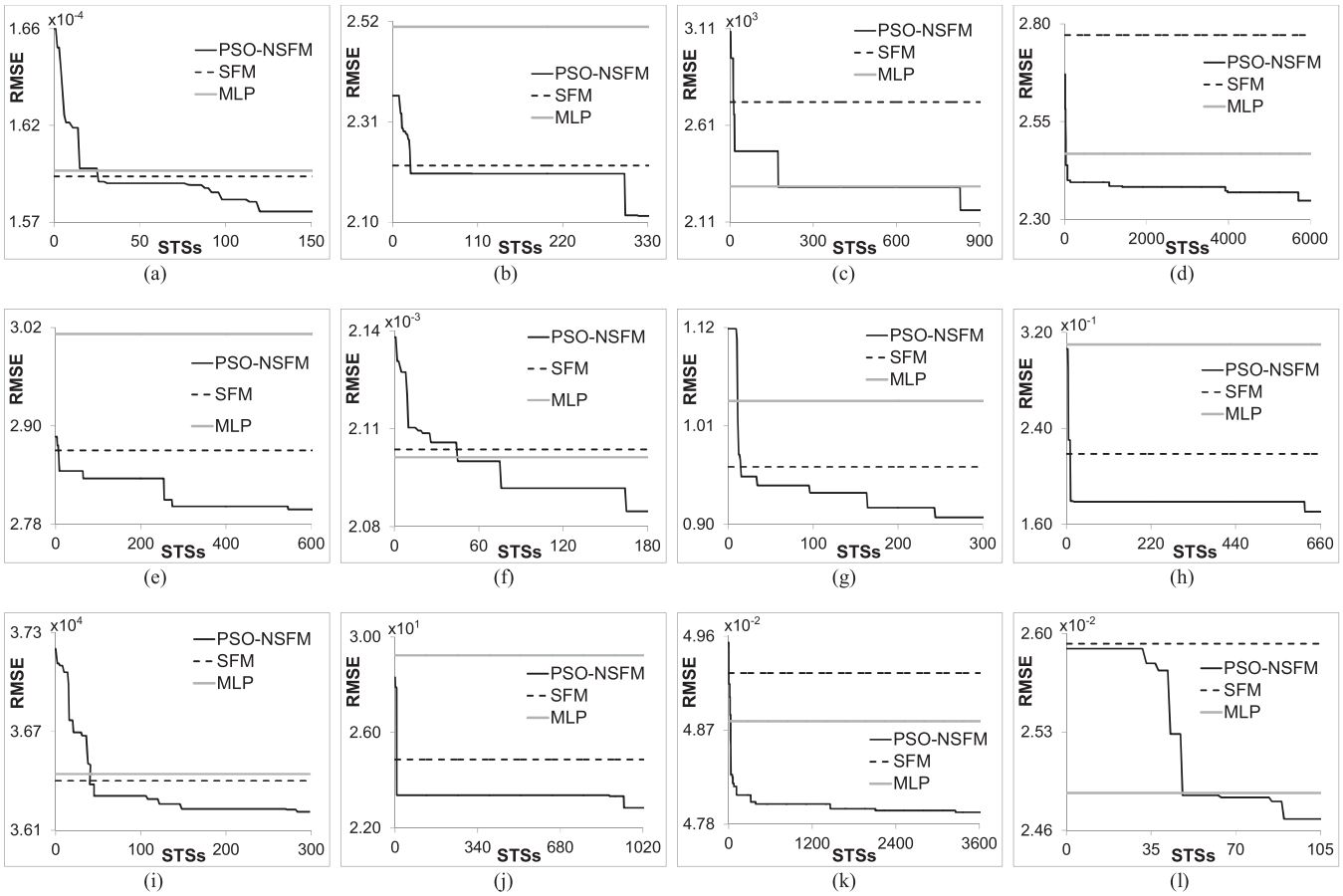

Fig. 4. Evolution of RMSE per STS in the validation set for the three methodologies. The best solution found in terms of RMSE in the validation dataset is depicted for the PSO-NSFM algorithm. (a) Ailerons. (b) Auto MPG. (c) Auto Price. (d) Boston Housing. (e) CPU Small. (f) Elevators. (g) Friedman. (h) Gas Furnace. (i) House Price-16H. (j) Machine CPU. (k) Mackey-Glass. (l) Samad.

number of nodes while still providing smaller approximation errors. In fact, in five cases the decrease in RBF centers selected by the PSO-NSFM algorithm exceeded 20%, compared to the number of centers selected by the SFM algorithm.

As far as simulation times are concerned, the PSO-NSFM algorithm requires more computational effort to arrive to completion compared to the SFM training method. This is expected because of the significant increase in the size of the search space that needs to be explored by the proposed algorithm. However, it is important to note that, within a few steps, the RBF model generated by the PSO-NSFM algorithm already outperforms the final outcome of SFM in terms of prediction accuracy. This is translated into computational times that are lower in all cases (considering either the best run or the average among the 30 runs), compared to the time needed by the SFM algorithm to arrive to completion. In fact, PSO-NSFM managed to outperform SFM right from the first time-step—as far as the best run is concerned—in five cases. Thus, the proposed approach generates a highly accurate model within the first time steps, but further improvements are being made until completion of the algorithm.

It should be noted that an increase in the number of examples and the problem dimensionality results in higher computational times. The increase in complexity could ultimately lead to computational times that are not practically feasible. However, the performance of the proposed approach on case studies with high dimensionality and large amounts of examples indicates that the algorithm can handle datasets with reasonably high complexity.

VI. Conclusion

The symmetric version of the FM algorithm provides a fast and reliable method for determining the hidden nodes of an RBF network. However, the produced RBF models could be improved both in terms of higher accuracy and parsimony by applying a different number of fuzzy sets in each input dimension. In this paper, we introduced a systematic approach for fully training RBF networks using a nonsymmetric fuzzy partitioning of the input space and PSO-based optimization of the number of fuzzy sets in each dimension. The produced methodology, namely, the PSO-NSFM algorithm, was evaluated on 12 synthetic and real-world benchmark datasets of various sizes. The resulting RBF models were found to exhibit smaller modeling errors in the validation and testing datasets compared to the original SFM algorithm and MLP networks. The higher accuracy produced by the PSO-NSFM-generated models was combined with a smaller number of hidden node centers in most of the tested cases. Furthermore, the computational times needed for the PSO-NSFM algorithm in order to outperform the original SFM algorithm in terms of modeling error were smaller compared to the total time needed for training the latter. The performance of the proposed approach on datasets with high complexity indicates that it could be used in image processing, speech recognition, biometrics, bioinformatics, and other computationally demanding tasks of practical interest.

Future research plans include substitution of the RBFs used in this paper with alternative forms of basis functions in order to further increase the model accuracy and decrease the computational load.

References

[1] C. A. Michelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Constr. Approx.*, vol. 2, pp. 11–22, Mar. 1986.

[2] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

[3] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.

[4] C. Darken and J. Moody, "Fast adaptive K-means clustering: Some empirical results," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, San Diego, CA, Mar. 1990, pp. 233–238.

[5] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[6] X. Hong, "A fast identification algorithm for box-cox transformation based radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 1064–1069, Jul. 2006.

[7] S. Chen, X. Hong, B. L. Luk, and C. J. Harris, "Construction of tunable radial basis function networks using orthogonal forward selection," *IEEE Trans. Syst., Man., Cybern. B, Cybern.*, vol. 39, no. 2, pp. 457–466, Apr. 2009.

[8] S. Chen, X. X. Wang, X. Hong, and C. J. Harris, "Kernel classifier construction using orthogonal forward selection and boosting with Fisher ratio class separability measure," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1652–1656, Nov. 2006.

[9] Y. W. Wong, K. P. Seng, and L. M. Ang, "Radial basis function neural network with incremental learning for face recognition," *IEEE Trans. Syst., Man., Cybern. B, Cybern.*, vol. 41, no. 4, pp. 940–949, Aug. 2011.

[10] M. Hou and X. Han, "Constructive approximation to multivariate function by decay RBF neural network," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1517–1523, Sep. 2010.

[11] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Comput.*, vol. 19, no. 2, pp. 461–478, 1997.

[12] A. Leonardis and H. Bischof, "An efficient MDL-based construction of RBF networks," *Neural Netw.*, vol. 11, no. 5, pp. 963–973, 1998.

[13] H. Sarimveis, A. Alexandridis, G. Tsekouras, and G. Bafas, "A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space," *Ind. Eng. Chem. Res.*, vol. 41, no. 4, pp. 751–759, 2002.

[14] W. Pedrycz, V. Loia, and S. Senatore, "Fuzzy clustering with viewpoints," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 274–284, Apr. 2010.

[15] U. Maulik, S. Bandyopadhyay, and I. Saha, "Integrating clustering and supervised learning for categorical data analysis," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 40, no. 4, pp. 664–675, Jul. 2010.

[16] X. X. Zhang, H. X. Li, and C. K. Qi, "Spatially constrained fuzzy-clustering-based sensor placement for spatiotemporal fuzzy-control system," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 5, pp. 946–957, Oct. 2010.

[17] A. Alexandridis, H. Sarimveis, and G. Bafas, "A new algorithm for online structure and parameter adaptation of RBF networks," *Neural Netw.*, vol. 16, no. 7, pp. 1003–1017, 2003.

[18] A. Alexandridis, H. Sarimveis, and K. Ninos, "A radial basis function network training algorithm using a non-symmetric partition of the input space—application to a model predictive control configuration," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 830–837, 2011.

[19] J. A. Leonard and M. A. Kramer, "Radial basis function networks for classifying process faults," *IEEE Control Syst.*, vol. 11, no. 3, pp. 31–38, Apr. 1991.

[20] M. Orr, "Optimising the widths of radial basis functions," in *Proc. 5th Brazilian Symp. Neural Netw.*, 1998, pp. 26–29.

[21] W. Yao, X. Chen, M. V. Tooren, and Y. Wei, "Euclidean distance and second derivative based widths optimization of radial basis function neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Barcelona, Spain, 2010, pp. 1–8.

[22] W. Yao, X. Chen, Y. Zhao, and M. V. Tooren, "Concurrent subspace width optimization method for RBF neural network modeling," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 247–259, Feb. 2012.

[23] S. F. Su, C. C. Chuang, C. W. Tao, J. J. Jeng, and C. C. Hsiao, "Radial basis function networks with linear interval regression weights for symbolic interval data," *IEEE Trans. Syst., Man., Cybern. B, Cybern.*, vol. 42, no. 1, pp. 69–80, Feb. 2012.

[24] D. Zhang, L. F. Deng, K. Y. Cai, and A. So, "Fuzzy nonlinear regression with fuzzified radial basis function network," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 6, pp. 742–760, Dec. 2005.

[25] R. Langari, L. Wang, and J. Yen, "Radial basis function networks, regression weights, and the expectation-maximization algorithm," *IEEE Trans. Syst., Man., Cybern. A, Syst. Humans*, vol. 27, no. 5, pp. 613–623, Sep. 1997.
[26] C. Andrieu, N. D. Freitas, and A. Doucet, "Robust full Bayesian learning for radial basis networks," *Neural Comput.*, vol. 13, no. 10, pp. 2359–2407, 2001.
[27] H. X. Huan, D. T. T. Hien, and H. H. Tue, "Efficient algorithm for training interpolation RBF networks with equally spaced nodes," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 982–988, Jun. 2011.
[28] T. Xie, H. Yu, J. Hewlett, P. Rózycki, and B. Wilamowski, "Fast and efficient second-order method for training radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 23, no. 4, pp. 609–619, Apr. 2012.
[29] H. Sarimveis, A. Alexandridis, S. Mazarakis, and G. Bafas, "A new algorithm for developing dynamic radial basis function neural network models based on genetic algorithms," *Comput. Chem. Eng.*, vol. 28, nos. 1–2, pp. 209–217, 2004.
[30] P. A. Gutiérrez, C. Hervás-Martínez, and F. J. Martínez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 246–263, Feb. 2011.
[31] J. González, I. Rojas, J. Ortega, H. Pomares, F. J. Fernández, and A. F. Díaz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1478–1495, Nov. 2003.
[32] O. Buchtala, M. Klimek, and B. Sick, "Evolutionary optimization of radial basis function classifiers for data mining applications," *IEEE Trans. Syst., Man., Cybern. B, Cybern.*, vol. 35, no. 5, pp. 928–947, Oct. 2005.
[33] F. Valdez, P. Melin, and O. Castillo, "Evolutionary method combining particle swarm optimisation and genetic algorithms using fuzzy logic for parameter adaptation and aggregation: The case neural network optimisation for face recognition," *Int. J. Artif. Intell. Soft Comput.*, vol. 2, nos. 1–2, pp. 77–102, 2010.
[34] F. Valdez, P. Melin, and O. Castillo, "An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2625–2632, 2011.
[35] S. Chen, X. Hong, and C. J. Harris, "Particle swarm optimization aided orthogonal forward regression for unified data modeling," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 477–499, Aug. 2010.
[36] D. S. Huang and J. X. Du, "A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2099–2115, Dec. 2008.
[37] S. K. Oh, W. D. Kim, W. Pedrycz, and S. C. Joo, "Design of K-means clustering-based polynomial radial basis function neural networks (pRBF NNs) realized with the aid of particle swarm optimization and differential evolution," *Neurocomputing*, vol. 78, no. 1, pp. 121–132, 2012.
[38] A. Alexandridis and H. Sarimveis, "Nonlinear adaptive model predictive control based on self-correcting neural network models," *AIChE J.*, vol. 51, no. 9, pp. 2495–2506, 2005.
[39] K. Ninos, C. Giannakakis, I. Kompogiannis, I. Stavrakas, and A. Alexandridis, "Nonlinear control of a DC-motor based on radial basis function neural networks," in *Proc. IEEE Int. Symp. Innov. Intell. Syst. Appl.*, Istanbul, Turkey, Jun. 2011, pp. 611–615.
[40] A. Alexandridis, P. Patrinos, H. Sarimveis, and G. Tsekouras, "A two-stage evolutionary algorithm for variable selection in the development of RBF neural network models," *Chemometrics Intell. Lab. Syst.*, vol. 75, no. 2, pp. 149–162, 2005.
[41] P. Patrinos, A. Alexandridis, K. Ninos, and H. Sarimveis, "Variable selection in nonlinear modeling based on RBF networks and evolutionary computation," *Int. J. Neural Syst.*, vol. 20, no. 5, pp. 365–379, 2010.
[42] A. Alexandridis, E. Chondrodima, K. Moutzouris, and D. Triantis, "A neural network approach for the prediction of the refractive index based on experimental data," *J. Mater. Sci.*, vol. 47, no. 2, pp. 883–891, 2012.
[43] A. Alexandridis, D. Triantis, I. Stavrakas, and C. Stergiopoulos, "A neural network approach for compressive strength prediction in cement-based materials through the study of pressure-stimulated electrical signals," *Construct. Build. Mater.*, vol. 30, pp. 294–300, May 2012.
[44] J. Nie, "Fuzzy control of multivariable nonlinear servomechanisms with explicit decoupling scheme," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 304–311, May 1997.
[45] A. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Chichester, U.K.: Wiley, 2007.
[46] A. Asuncion and D. J. Newman. (2007). *UCI Machine Learning Repository*. Univ. California, Irvine [Online]. Available: http://archive.ics.uci.edu/ml/
[47] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden-Day, 1970.
[48] K. Kilic, B. A. Sproule, I. B. Türksen, and C. A. Naranjo, "A fuzzy system modeling algorithm for data analysis and approximate reasoning," *Robot. Auton. Syst.*, vol. 49, nos. 3–4, pp. 173–180, 2004.
[49] J. Friedman, "Multivariate adaptive regression splines," *Ann. Stat.*, vol. 19, no. 1, pp. 1–67, 1991.
[50] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 359–372, Mar. 2007.
[51] C. C. Holmes and B. K. Mallick, "Bayesian wavelet networks for nonparametric regression," *IEEE Trans. Neural Netw.*, vol. 11, no. 1, pp. 27–35, Jan. 2000.
[52] C. K. Loo, M. Rajeswari, and M. V. C. Rao, "Novel direct and self-regulating approaches to determine optimum growing multi-experts network structure," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1378–1395, Nov. 2004.
[53] T. Samad, "Backpropagation with expected source values," *Neural Netw.*, vol. 4, no. 5, pp. 615–618, 1991.
[54] M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

Alex Alexandridis (M'12) received the Diploma degree in chemical engineering and the Ph.D. degree in computational intelligence and control from the National Technical University of Athens (NTUA), Athens, Greece, in 2000 and 2003, respectively.

He was a Post-Doctoral Fellow with NTUA from 2004 to 2009. Since 2010, he has been an Assistant Professor with the Department of Electronics, Technological Educational Institute of Athens, Athens, Greece heading the Computational Intelligence Unit. He has authored or co-authored more than 50 original research works. His current research interests include computational intelligence, nonlinear system modeling and control with emphasis on model predictive control methods, intelligent control, and applications to process engineering, materials science, and environmental science.

Eva Chondrodima received the B.Sc. degree in electronic engineering from the Technological Educational Institute of Athens, Athens, Greece, in April 2011. She is currently pursuing the M.Sc. degree in advanced information systems with the Department of Informatics, University of Piraeus, Piraeus, Greece.

Her current research interests include neural networks, evolutionary computation methods, and decision support systems.

Haralambos Sarimveis received the Diploma degree in chemical engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 1990, and the M.Sc. and Ph.D. degrees in chemical engineering from Texas A&M University, College Station, in 1992 and 1995, respectively.

He has been with the School of Chemical Engineering at NTUA since 2000. Currently, he is an Associate Professor, heading the Process Control and Informatics Laboratory. His research work has resulted in more than 80 articles in scientific journals and a large number of papers in scientific conferences. His current research interests include analysis and identification of dynamical systems, automatic control with emphasis on model predictive control, computational intelligence, computer aided molecular design, and supply chain management.