Advances in Engineering Software 42 (2011) 830–837

Contents lists available at ScienceDirect

Advances in Engineering Software

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

A Radial Basis Function network training algorithm using a non-symmetric partition of the input space – Application to a Model Predictive Control configuration

Alex Alexandridis [a,*], Haralambos Sarimveis [b], Konstantinos Ninos [b]

[a] Department of Electronics, Technological Educational Institute of Athens, Agiou Spiridonos, Aigaleo 12210, Greece
[b] School of Chemical Engineering, National Technical University of Athens, 9 Heroon Polytechniou Street, Zografou Campus, Athens 15780, Greece

ARTICLE INFO

Article history:
Received 8 January 2009
Received in revised form 23 April 2010
Accepted 17 May 2011
Available online 12 June 2011

Keywords:
Neural networks
Radial basis function
Fuzzy means
Fuzzy partition
Non-symmetric partition
Model predictive control
Mackey-Glass
Continuous stirred tank reactor

ABSTRACT

This work presents the non-symmetric fuzzy means algorithm which is a new methodology for training Radial Basis Function neural network models. The method is based on a non-symmetric fuzzy partition of the space of input variables which results to networks with smaller structures and better approximation capabilities compared to other state-of-the-art training procedures. The lower modeling error and the smaller size of the produced models become particularly important when they are used in online applications. This is demonstrated by integrating the model produced by the proposed algorithm in a Model Predictive Control configuration, resulting in better control performance and shorter computational times.

1. Introduction

Radial Basis Function (RBF) neural networks constitute a special neural network architecture that has received much attention from the academic community. A typical RBF network comprises only one hidden layer of neurons which results to fewer synaptic weights and in general a less complex structure compared to other neural network architectures. Typically there are two approaches for training an RBF network: The first approach aims at determining all the network parameters in one step using a nonlinear optimization procedure. Following this approach, Sarimveis et al. [19] developed a special genetic algorithm to auto-configure the structure of the network and obtain the model parameters. Hefny et al. [6] introduced a fuzzy neural network which can be considered as a logical version of RBF networks, where genetic algorithms are employed as the learning mechanism. Peng et al. [17] proposed a hybrid forward algorithm for the construction of RBF neural networks with tunable nodes, which according to the authors leads to an improved network performance and reduced memory usage for the network construction. Wedge et al. [22] presented a hybrid RBF-sigmoid neural network with a three-step training algorithm that utilizes both global search and gradient descent training. Their algorithm is intended to identify global features of an input–output relationship before adding local detail to the approximating function. As an alternative to these gradient-based procedures, but still calculating the network parameters in one step, Lazaro et al. [10] used the Expectation–Maximization algorithm to obtain maximum likelihood estimates of the RBF network parameters.

The second approach for training an RBF network is to separate the problem of identifying the network parameters in two steps: The first step aims at finding the number and locations of the hidden node RBF centers, while in the second step the synaptic weights are determined. As the second step is performed trivially by linear regression between the outputs of the hidden node and the real output data, this two-step procedure is usually faster than optimizing all the RBF network parameters at the same time.

Still, determining the number and the locations of the hidden node centers is not an easy task and numerous attempts to solve this problem have been presented in the literature. Among the most recent ones, Panchapakesan et al. [14] proposed a methodology where the locations of the centers are selected using unsupervised methods and then are fined-tuned to decrease the training error. Jiang et al. [8] used a genetic algorithm to select the

* Corresponding author. Address: Department of Electronics, Technological Educational Institute of Athens, Agiou Spiridonos, Aigaleo 12210, Greece. Tel.: +30 2105358358.
E-mail address: alexx@teiath.gr (A. Alexandridis).

0965-9978/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2011.05.026

appropriate network structure in determining the optimal number of nodes. Zhao and Huang [20] introduced an evolutionary structure optimization method for selecting the RBF hidden centers and widths. Sarimveis et al. [18] proposed the fuzzy means algorithm which determines the RBF centers of the network based on a fuzzy partition of the input space; later an adaptive version of the algorithm was proposed for modeling time-varying systems [1]. The main advantage of the algorithm is that it has the ability to determine both the centers and structure of the network in very short computational times, while comparisons with other training methodologies show that the prediction capabilities of the produced models are similar or superior.

The fuzzy means algorithm selects the hidden node centers by partitioning the input space to an equal number of fuzzy sets for each input variable. However, it might be possible to improve the results in terms of prediction accuracy and/or network size, by assigning a different number of fuzzy sets to each input variable. The modification of the original method in order to take into account non-symmetric fuzzy partitions of the input space is not trivial. The goal of this paper is to investigate this possibility by using hyper-ellipsoid fuzzy subspaces instead of the hyper-spherical shapes on which the original algorithm was based. Also the concept of the relative Euclidean distance introduced by Nie [13], is extended to account for the non-symmetric partition.

The main result of this study is the development of the non-symmetric fuzzy means algorithm which is a new training. The algorithm improves the prediction accuracy of the produced models, while at the same time a significant reduction of the number of hidden nodes is achieved. Reduced complexity is a desired model feature, especially when the model is used for real time applications. In order to demonstrate the benefits of the non-symmetric algorithm, the proposed method is combined with a standard Model Predictive Control (MPC) configuration. MPC algorithms make use of a model of the controlled process in order to estimate the optimum sequence of control moves that will drive the process closer to the set-point [7]. Incorporating RBF dynamical models in such a control configuration is not a new concept. Peng et al. [15,16] presented a methodology for training RBF-based AutoRegressive models with eXogenuous (ARX) variables and then implemented it in the design of an MPC for nonlinear industrial processes. In Alexandridis and Sarimveis [2], an adaptive nonlinear control scheme was introduced, by integrating the symmetric adaptive fuzzy means algorithm into the MPC framework. The complete control scheme was then applied successfully to a digester reactor. The impact of the network size on the computational time needed for solving the online optimization problem was not investigated in details, since for the particular application there was sufficient time between taking two consecutive control actions. However in applications with fast dynamics, the computational time for solving the optimization problem is critical. In such cases it is necessary to find ways to speed up the optimization task. Bhartiya and Whiteley [3,4] proposed an MPC approach, where a factorized RBF network serves as the process model. In their work the factorization of the RBF network increases the computational efficiency of the control algorithm thus helping to solve the optimization problem faster. Wang et al. [21] applied an MPC methodology based on RBF networks for the adaptive control of the air-fuel ratio in a spark ignition engine. Using a reduced Hessian method, they managed to speed up the nonlinear optimization problem. In this paper we show that utilization of the non-symmetric fuzzy means algorithm results to a model with better accuracy, which guarantees a better control performance, but also decreases the computational burden for solving the on-line MPC optimization problem. This result is verified by applying the proposed MPC configuration to a Continuous Stirred Tank Reactor (CSTR) model.

The rest of this article is organized as follows: A short introduction to the general concept of the fuzzy means algorithm is given in the next section, followed by the presentation of the proposed non-symmetric modification. Next follows a description of how RBF models can be integrated within the MPC configuration and an analysis of the computational complexity of the resulting optimization problem. Then, two case studies are presented, where the proposed methodology is applied to benchmark modeling and control problems and compared with the original version of the fuzzy means algorithm. In the final section we draw conclusions and set some directions for future research.

2. The fuzzy means algorithm

The fuzzy means algorithm has been proposed as an alternative to classical methodologies for selecting the RBF network hidden node centers, like the *k*-means algorithm [5,12]. In contrast to the traditional methodologies, the fuzzy means algorithm has the ability to determine automatically the size of the network, i.e. the number of RBF centers, while it proves to be orders of magnitude faster. The main idea behind the algorithm is the partition of the input space into a number of fuzzy sets which is described next.

Consider a system with N normalized input variables x_i, where $i = 1, \ldots, N$. The case of symmetric partition will be investigated first, where the domain of each input variable is partitioned into an equal number of one-dimensional triangular fuzzy sets, c. Each fuzzy set can be written as:

$$A_{i,j} = \{a_{i,j}, \delta\alpha\}, \quad i = 1, \ldots, N,\ j = 1, \ldots, c \tag{1}$$

where $a_{i,j}$ is the center element of fuzzy set $A_{i,j}$ and $\delta\alpha$ is half of the respective width (due to the symmetric partition all the widths are equal). This partitioning technique creates a total of c^N multi-dimensional fuzzy subspaces $\mathbf{A}^l$, where $l = 1, \ldots, c^N$. Each multi-dimensional fuzzy subspace is generated by combining N one-dimensional fuzzy sets, one for each input direction. One can define the center vector $\mathbf{a}^l$ and the side vector $\delta\mathbf{a}$ of each fuzzy subspace:

$$\mathbf{A}^l = \{\boldsymbol{\alpha}^l, \delta\boldsymbol{\alpha}\} = \left\{ \left[a^l_{1,j_1}, a^l_{2,j_2}, \ldots, a^l_{N,j_N}\right], \underbrace{[\delta\alpha, \delta\alpha, \ldots, \delta\alpha]}_{N} \right\}, \quad l = 1, \ldots, c^N \tag{2}$$

where a^l_{i,j_i} is the center element of the one-dimensional fuzzy set A_{i,j_i} that has been assigned to input i. Each one of the produced fuzzy subspaces is a candidate for becoming an RBF center but only a few of those will be finally selected. The selection is based on the idea of the multidimensional membership function $\mu_{A^l}(\mathbf{x}(k))$ of an input vector $\mathbf{x}(k)$ to a fuzzy subspace $\mathbf{A}^l$ which is given by Nie [13]:

$$\mu_{A^l}(\mathbf{x}(k)) = \begin{cases} 1 - rd^l(\mathbf{x}(k)), & \text{if } rd^l(\mathbf{x}(k)) \leqslant 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $rd^l(\mathbf{x}(k))$ is the Euclidean relative distance between $\mathbf{A}^l$ and the input data vector $\mathbf{x}(k)$:

$$rd^l(\mathbf{x}(k)) = \sqrt{\sum_{i=1}^{N} \left(a^l_{i,j_i} - x_i(k)\right)^2} \Big/ \sqrt{N}\delta a \tag{4}$$

Eq. (4) defines a hyper-sphere on the input space with radius equal to $\sqrt{N}\delta a$. The objective of the training algorithm is to select a subset of fuzzy subspaces as RBF centers so that all the training data are covered by at least one hyper-sphere. Expressing this requirement in terms of Eq. (3), the subset of fuzzy subspaces is selected so that there is at least one fuzzy subspace that assigns a nonzero multidimensional degree to each input training vector. The algorithm that

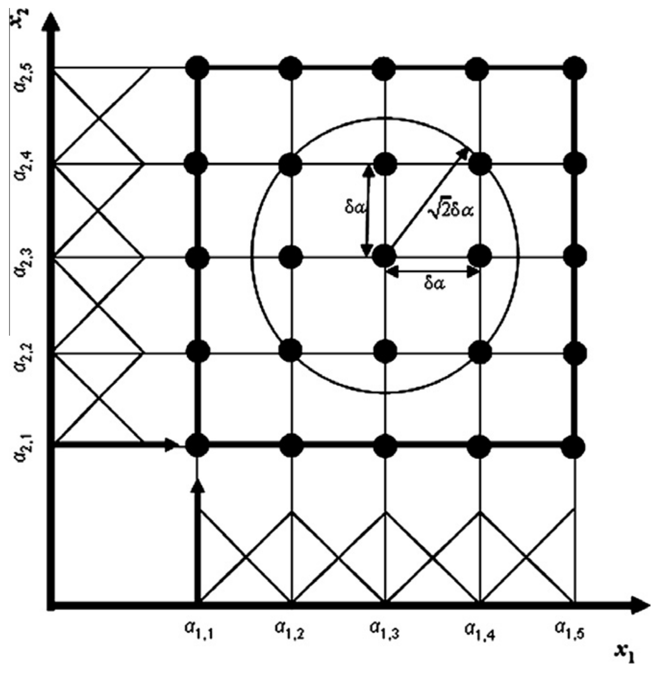

Fig. 1a. Two dimensional example: Symmetric fuzzy partitioning into five fuzzy sets for both input variables.

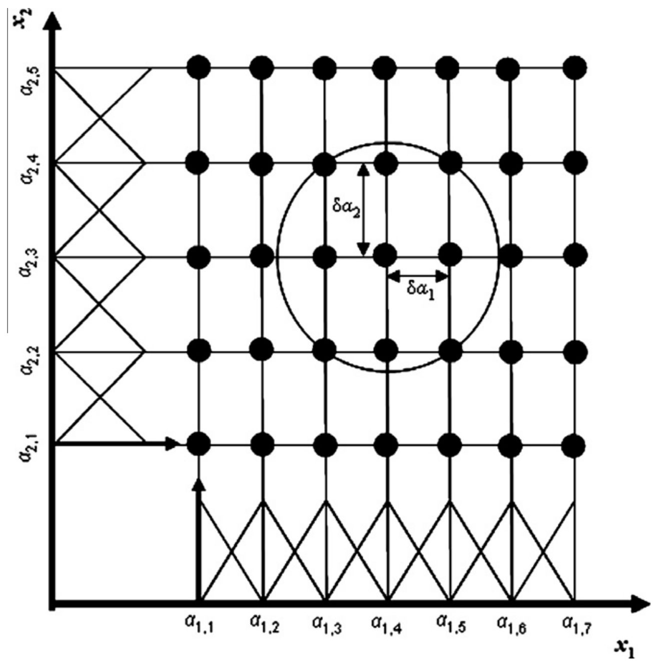

Fig. 1b. Two dimensional example: Non-symmetric fuzzy partitioning into seven fuzzy sets for x_1 and five fuzzy sets for x_2.

is responsible for the selection of this subset is described in [18], followed by an analysis of its low computational complexity. More details about fuzzy partitioning can be found in the above publication.

An example of this type of fuzzy partitioning can be seen in Fig. 1a, where for visualization purposes a two-dimensional input space is depicted. The domains of the two input variables x_1, x_2 are partitioned into five fuzzy sets, thus creating 16 rectangles. Obviously this symmetric partitioning procedure leads to a square grid, where the Euclidean relative distance defines a circle with radius equal to $\sqrt{2}\delta a$. Using more fuzzy sets for each input variable makes the grid denser and results to the selection of more RBF centers. The distribution of the candidate centers along the direction of each input variable is the same as long as the same number of fuzzy sets is used for all of them. However this restricts the flexibility of the algorithm, since a different partitioning for each input variable might result in a better network in terms of accuracy and/or complexity of the model. In the next subsection, the necessary modifications are presented in order to extend the applicability of the algorithm to a non-symmetric partitioning of the input space.

2.1. Non-symmetric fuzzy means

In the case of non-symmetric fuzzy partition, the domain of each input variable is partitioned into c_i fuzzy sets where $i = 1, \ldots, N$ which results to C fuzzy subspaces:

$$C = \prod_{i=1}^{N} c_i \tag{5}$$

The widths of the fuzzy sets are different for each input variable, and the fuzzy subspaces are described as follows:

$$\mathbf{A}^l = \{\boldsymbol{\alpha}^l, \delta\boldsymbol{\alpha}\} = \left\{\left[a^l_{1,j_1}, a^l_{2,j_2}, \ldots, a^l_{N,j_N}\right], [\delta\alpha_1, \delta\alpha_2, \ldots, \delta\alpha_N]\right\},$$
$$l = 1, \ldots, C \tag{6}$$

An example of the non-symmetric partition is displayed in Fig. 1b using again a two-dimensional case. Now, the domain of input variable x_1 is partitioned into seven fuzzy sets while the domain of x_2 is partitioned into five fuzzy sets. The grid defines 24 equal rectangles in the input space and the two edges of each rectangle are twice the widths of the respective fuzzy sets $\delta\alpha_1$, $\delta\alpha_2$.

The definitions of the Euclidean relative distance and the multi-dimensional membership function in Eqs. (3) and (4) are suitable only for the symmetric partition. In the sequel we will examine how these definitions can be modified to account for the non-symmetric partition. In particular, we will associate an N-dimensional hyper-ellipse with each fuzzy subspace:

$$\sum_{i=1}^{N} \frac{(a^l_{i,j_i} - x_i(k))^2}{(b_i)^2} = 1, \quad l = 1, \ldots, C \tag{7}$$

where b_i is half of the respective ellipse axis for each variable $i = 1, \ldots, N$. The parameters b_i need to be determined, in contrast to the hyper-sphere for which all the parameters b_i $i = 1, \ldots, N$ are equal to 1. A first prerequisite is that the surface of the hyper-ellipse needs to cross all the vertices of the hyper-rectangle defined by the fuzzy subspace e.g. in the two-dimensional example it should cross the four vertices of the rectangle. However this condition is not sufficient to define uniquely the hyper-ellipse since there are infinite hyper-ellipses that cross all the vertices – a special case of which is the hyper-sphere of Eq. (4). Among them, we select the one whose axes are proportional to the edges of the hyper-rectangle, i.e.:

$$\frac{\delta a_1}{b_1} = \frac{\delta a_2}{b_2} = \cdots = \frac{\delta a_N}{b_N} \tag{8}$$

The inclusion of Eq. (8) in the formulation of the problem defines uniquely the hyper-ellipse. It can easily be derived that this is described by the following equation:

$$\sum_{i=1}^{N} \left(\left(a^l_{i,j_i} - x_i(k)\right)^2 / N(\delta a_i)^2\right) = 1 \tag{9}$$

Fig. 1c depicts the ellipse that is finally formulated in the two-dimensional case.

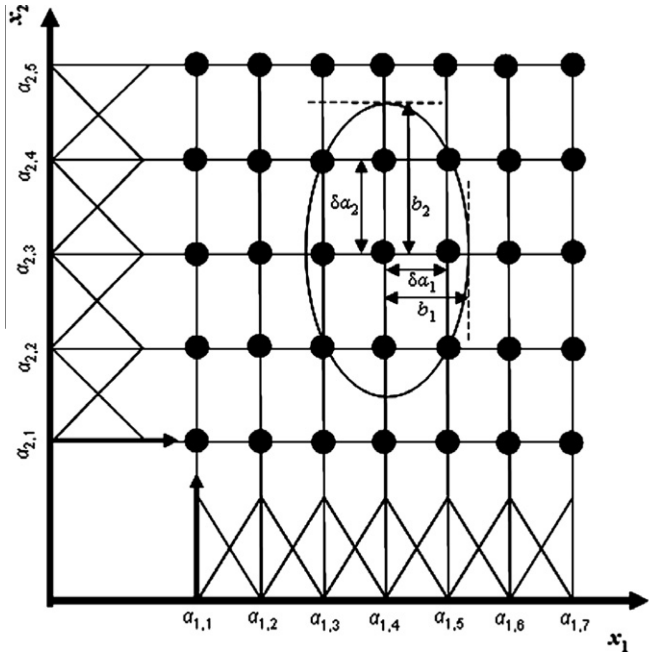

Fig. 1c. Two dimensional example: Ellipsoid shaped membership function for non-symmetric fuzzy partitioning.

Based on Eq. (9), a new multidimensional membership function is derived where the relative distance between a fuzzy subspace $\mathbf{A}^l$ and the input data vector $\mathbf{x}(k)$ is given by:

$$rd^l(\mathbf{x}(k)) = \sqrt{\sum_{i=1}^{N}\left(\left(a^l_{i,j_i} - x_i(k)\right)^2 / N(\delta a_i)^2\right)} \qquad (10)$$

The replacement of the original spherical relative distance equation with an ellipsoidal one makes the non-symmetric algorithm more flexible, as it gives the user the opportunity to partition the domain of each input variable in a different way. Thus the resulting RBFs cover more efficiently the regions of the input space where input data are available and produce networks with more accuracy (smaller modeling error) and lower complexity.

After replacing the relative distance equation, the algorithm proceeds in the same way with the original fuzzy means methodology to find the subset of fuzzy subspaces that assign a nonzero multidimensional degree to all input training vectors.

3. Model Predictive Control using Radial Basis Function models

Typically, MPC controllers make use of a model correlating the controlled variable with the manipulated one. Then at each discrete time step an optimization problem is formulated, where the objective is to minimize the difference between the set point and the predictions of the model. The solution to this problem is the optimum sequence of control moves that drives the controlled variable to the set point value. A graphical representation of the MPC methodology is shown in Fig. 2.

In a typical MPC implementation, the objective function is a combination of two targets: (a) minimization of the distances between the predicted output values and the set point and (b) minimization of the control moves:

$$\min_{\Delta\mathbf{v}(k),\Delta\mathbf{v}(k+1),\ldots\Delta\mathbf{v}(k+h_c)} \sum_{i=1}^{h_p}\|\mathbf{\Theta}(\hat{\mathbf{y}}(k+i) - y^{sp})\|_2^2 + \sum_{i=0}^{h_c}\|\mathbf{\Omega}\Delta\mathbf{v}(k+i)\|_2^2 \qquad (11)$$

where $\Delta\mathbf{v}(k)$ is the vector containing the control moves at time step k; $\mathbf{\Theta}$ and $\mathbf{\Omega}$ are the error and move suppression weights; h_c and h_p

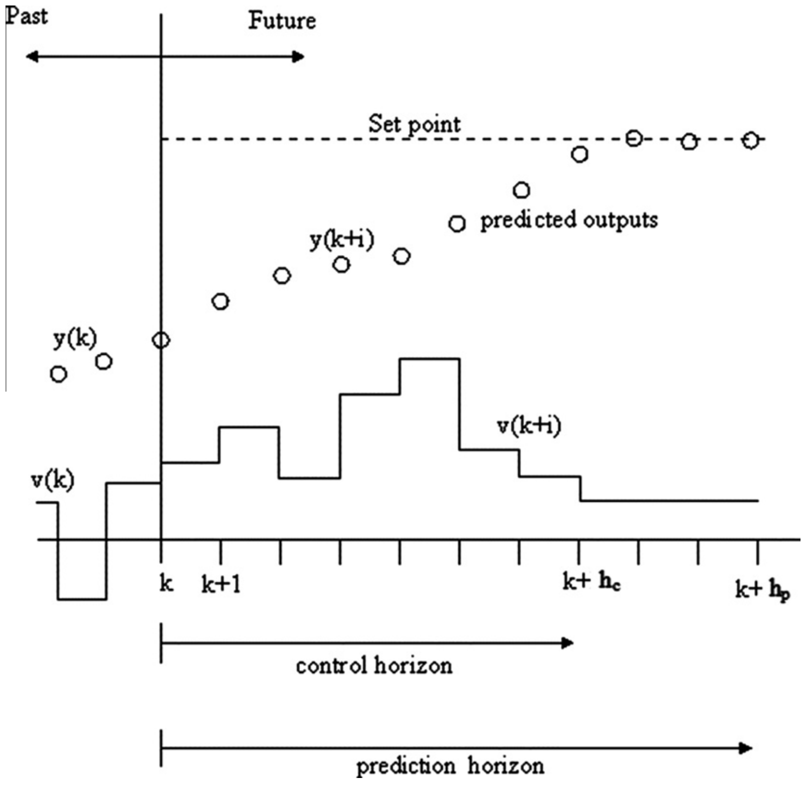

Fig. 2. A graphical representation of the MPC methodology.

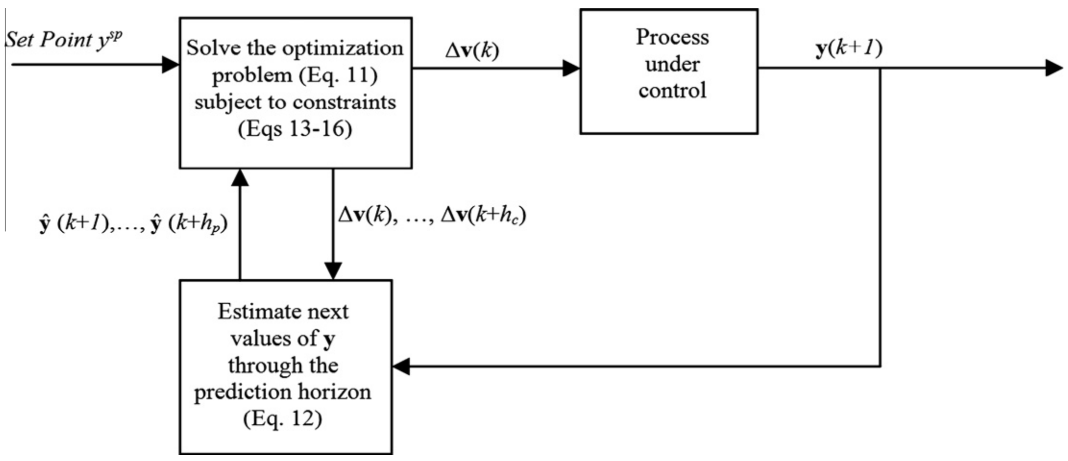
Set Point y^{sp}
Solve the optimization problem (Eq. 11) subject to constraints (Eqs 13-16)
$\Delta\mathbf{v}(k)$
Process under control
$\mathbf{y}(k+1)$
$\hat{\mathbf{y}}(k+1),\ldots,\hat{\mathbf{y}}(k+h_p)$
$\Delta\mathbf{v}(k),\ldots,\Delta\mathbf{v}(k+h_c)$
Estimate next values of $\mathbf{y}$ through the prediction horizon (Eq. 12)

Fig. 3. Integration of RBF model and MPC methodology.

Table 1
Parameters for the MacKey–Glass equation.

Parameter	Value
a	0.2
b	0.1
τ	30

Table 2
Case Study I: Results on the validation dataset for the symmetric and non-symmetric fuzzy means algorithm.

	Symmetric FM	Non-symmetric FM
Fuzzy partition	[16 16 16 16 16 16]	[9 11 13 9 13 13]
# of RBF centers	20	17
MARE%	0.46	0.23

are the control and prediction horizons respectively and $\mathbf{y}^{sp}$ is the set point value.

The minimization problem is solved subject to a number of constraints:

$$\hat{\mathbf{y}}(k+i) = \mathbf{NN}(k+i) + \mathbf{E}(k), \quad 1 \leqslant i \leqslant h_p \tag{12}$$

$$\mathbf{E}(k) = \mathbf{y}(k) - \mathbf{NN}(k) \tag{13}$$

$$\mathbf{v}_{\min} \leqslant \mathbf{v}(k+i) \leqslant \mathbf{v}_{\max}, \quad 1 \leqslant i \leqslant h_c \tag{14}$$

$$-\Delta\mathbf{v}_{\max} \leqslant \Delta\mathbf{v}(k+i) \leqslant \Delta\mathbf{v}_{\max}, \quad 1 \leqslant i \leqslant h_c \tag{15}$$

$$\Delta\mathbf{v}(k+i) = 0, \quad h_c + 1 \leqslant i \leqslant h_p \tag{16}$$

where $\mathbf{NN}(k)$ is the RBF network prediction for the time step k and $\mathbf{E}(k)$ is the current error between the actual output measurement and the model prediction. Eqs. (12) and (13) denote that the modeling error measured at time point k is assumed to be the same over the entire prediction horizon. Eqs. (14) and (15) pose upper and lower bounds on the values of the manipulated variables and the control moves, while Eq. (16) denotes that no control moves are allowed after the end of the control horizon. Fig. 3 shows schematically how the RBF model and the MPC methodology are integrated.

3.1. Solving the optimization problem – computational complexity

The nonlinear optimization problem defined by Eqs. (11)–(16) is solved using the sequential quadratic programming (SQP) algorithm included in the *fmincon* MATLAB function. SQP algorithms involve the calculation of the cost function (Eq. (11)) several times during each iteration. The most time consuming part in this calculation is computation of the RBF model response (Eq. (12)). The computational burden of the optimization problem increases with the number of input variables N, the number of times the cost function is called at each time step k and the number of centers in the RBF model. The importance of reducing the number of hidden node centers becomes apparent, especially for large scale models involving many input variables.

4. Case Study I: application of the new methodology to the Mackey–Glass time series

The proposed non-symmetric algorithm was tested by applying it to the prediction of the well-known Mackey–Glass time series [11]. The Mackey–Glass is a chaotic time series that has been used extensively for evaluating neural network models; for the discrete case, it arises from the following difference equation:

$$x(t+1) = (1-b)x(t) + a\frac{x(t-\tau)}{(1+x^{10}(t-\tau))} \tag{17}$$

The objective here is to build a model that predicts the next value for the Mackey–Glass time series, based on the six previous ones:

$$x(t) = \mathrm{RBF}(x(t-1), x(t-2), \ldots, x(t-6)) \tag{18}$$

Using the values of the parameters shown in Table 1, a set of 4000 values was generated. The first 1000 were discarded, whereas the next 1500 were used for training and the rest for validating the produced RBF models. For comparison purposes the original symmetric fuzzy means algorithm was applied to partitions from 4 to 20 fuzzy sets for all input variables. In the case of the non-symmetric algorithm, an exhaustive search was also performed testing all combinations of partitions ranging from 4 to 20 fuzzy sets for each input variable. The best results for both algorithms are presented in Table 2, where it can be seen that the non-symmetric algorithm outperforms the symmetric one in terms of smaller Mean Absolute Relative Error% (MARE%) and fewer RBF centers.

5. Case Study II: MPC of a Continuous Stirred Tank Reactor (CSTR)

This section presents an example where the non-symmetric fuzzy means algorithm is integrated as a modeling algorithm to a typical MPC configuration aiming to control a simulated system. The simulation involves a nonisothermal CSTR where the following exothermal irreversible reaction between sodium thiosulfate and hydrogen peroxide is taking place:

$$2Na_2S_2O_3 + 4H_2O_2 \rightarrow Na_2S_3O_6 + Na_2SO_4 + 4H_2O \tag{19}$$

This process is described by the following mass and energy balances [9]:

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F}{V}(C_{A,in} - C_A) - 2k_o \exp\left(-\frac{E}{RT}\right)C_A^2 \\ \frac{dT}{dt} &= \frac{F}{V}(T_{in} - T) + 2\frac{(-\Delta H)_R}{\rho c_p}k_o \exp\left(-\frac{E}{RT}\right)C_A^2 - \frac{UA}{V_{\rho c_p}}(T - T_j) \end{aligned} \tag{20}$$

where V is the volume of the CSTR; $(\Delta H)_R$ is the heat of the reaction; $-E/R$, k_o, c_p, ρ are constants of the reaction and the reactants; F is the flow rate into the reactor; $C_{A,in}$ is the inlet concentration of the reactant $Na_2S_2O_3$; C_A is the concentration of $Na_2S_2O_3$ inside the reactor; T_{in} is the inlet temperature; T is the temperature inside the reactor; T_j is the temperature of the coolant and UA is the overall heat-transfer coefficient between the cooling coil and the reactor contents. The values of the process parameters are shown in Table 3.

It should be noted that this type of CSTR has normally three steady state points, an upper and a lower one which are stable, and a medium one which is unstable. Though it is relatively easy to control the CSTR around the upper or lower steady state point, controlling the CSTR over the whole operating range which includes the unstable steady state point is a rather challenging task.

The CSTR was simulated by solving the system of ODEs in Eq. (20). The objective was to apply an MPC configuration in order to control concentration C_A using the temperature of the coolant T_j as the manipulated variable. The MPC configuration requires a dynamic model that correlates the output variable C_A with the input variable T_j. The presence of multiple steady states makes it impossible to approximate the system dynamics over the whole operating region using a model of the type:

$$C_A(k) = \text{RBF}(T_j(k-1), T_j(k-2), \ldots, T_j(k-i)) \tag{21}$$

i.e. a model using as inputs only past values of T_j. The reason is that for the same sequence of inputs $T_j(k-1)$, $T_j(k-2), \ldots, T_j(k-i)$, there are more than one possible values of the output variable $C_A(k)$, depending on which steady state the CSTR is nearest at that time point.

In order to circumvent this problem, an ARX (AutoRegressive with eXogenous inputs) type model was chosen, where the current concentration $C_A(k)$ is correlated with the previous value of the coolant temperature $T_j(k-1)$, but also with the previous values of the two state variables, i.e. the concentration $C_A(k-1)$ and the temperature inside the reactor $T(k-1)$:

$$C_A(k) = \text{RBF}(T_j(k-1), C_A(k-1), T(k-1)) \tag{22}$$

Table 3
Process parameter values for the CSTR.

Process parameter	Value
$\underline{V}$	100 l
UA	20,000 J/s K
Ρ	1000 g/l
C_p	4.2 J/g K
$-(\Delta H)_R$	596,619 J/mol
k_o	6.85E+11 l/s mol
E	76534.704 J/mol
R	8.314 J/mol K
F	20 l/s
T_j	275 K
$C_{A,in}$	1 mol/l

Table 4
Case Study II: Results on the validation datasets for the symmetric and non-symmetric fuzzy means algorithm.

	C_A model		T model	
	Symmetric FM	Non-symmetric FM	Symmetric FM	Non-symmetric FM
Fuzzy partition	[14 14 14]	[12 8 9]	[14 14 14]	[6 15 18]
# of RBF centers	76	37	76	46
MARE%	0.84	0.53	0.16	0.12

The introduction of such a model – using the previous values of the two state variables as inputs – complicates the calculation of the future model predictions throughout the prediction horizon. At each discrete time step k, where the optimization problem (Eq. (11)) is formulated and solved, the model (Eq. (22)) is used to generate predictions for the next h_p time steps. However in order to calculate the predictions beyond the first future time instance, e.g. $C_A(k+2)$, the values of both state variables during the previous time step $C_A(k+1)$ and $T(k+1)$ are needed as inputs to the model. Therefore, an additional model is needed to predict the dynamic evolution of the second state variable:

$$T(k) = \text{RBF}(T_j(k-1), C_A(k-1), T(k-1)) \tag{23}$$

In order to generate data for training the two RBF models, the CSTR was excited by changing the coolant temperature T_j every 1 s, within the limits 0–500. Using the described configuration, 50,000 data points were collected from the CSTR. The data points were split into a training dataset of 35,000 data points and a validation one of 15,000 data points. For comparison purposes, both the original version of the symmetric fuzzy means and the non-symmetric extension were applied. As in the Mackey–Glass example, the original symmetric fuzzy means algorithm was applied to partitions from 4 to 20 fuzzy sets for all input variables, while for the non-symmetric algorithm, an exhaustive search was performed testing all combinations of partitions ranging from 4 to 20 fuzzy sets for each input variable. Table 4 presents the best results obtained using the symmetric and the non-symmetric algorithms. It can be seen that the best networks produced by the non-symmetric algorithm outperform the ones generated by the symmetric fuzzy partition in terms of lower MARE%. Moreover, the lower modeling error is accompanied by a significant decrease in the number of RBF centers. To be more specific, the RBF networks produced by the non-symmetric algorithm contain only 37 and 46 centers for the C_A and T models respectively (which corresponds to a reduction of 51% and 39% respectively, compared to the best networks trained with a symmetric fuzzy partition).

In order to illustrate the combined benefits of the smallest modeling error and the reduction in RBF centers, the models trained with the two methodologies were incorporated into the MPC configuration described in the previous section, resulting in two different control schemes. The values of the operational parameters used by the MPC schemes are shown in Table 5. The controllers were

Table 5
Operational parameters used by both MPC schemes.

Controller parameter	Value
$\mathbf{\Omega}$	1
$\mathbf{\Theta}$	700
h_c	4
h_p	12
$\mathbf{v}_{\min}$	150
$\mathbf{v}_{\max}$	350
$\Delta\mathbf{v}_{\max}$	100

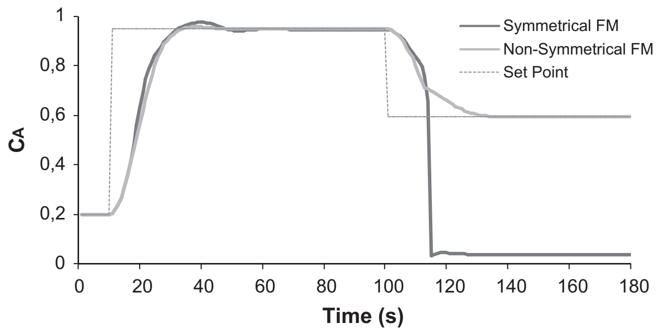

Fig. 4a. Case Study II, first case: Responses for the two MPC schemes.

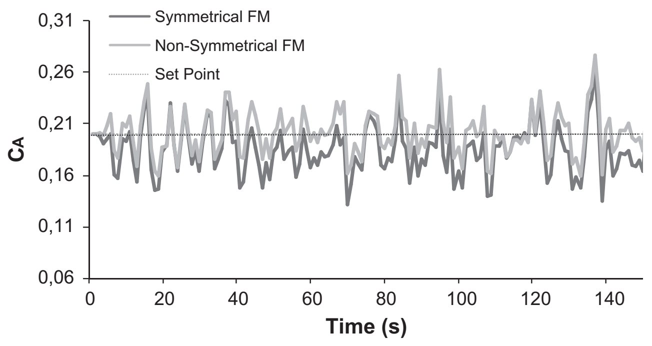

Fig. 5a. Case Study II, second case: Responses for the two MPC schemes.

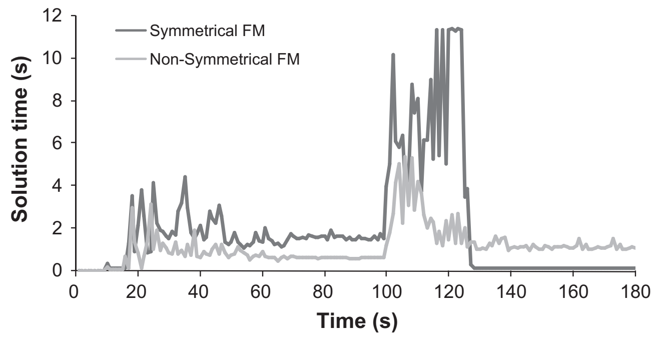

Fig. 4b. Case Study II, first case: Solution time for the two MPC schemes.

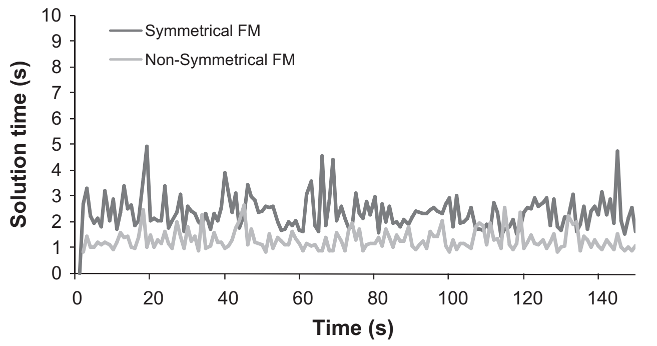

Fig. 5b. Case Study II, second case: Solution time for the two MPC schemes.

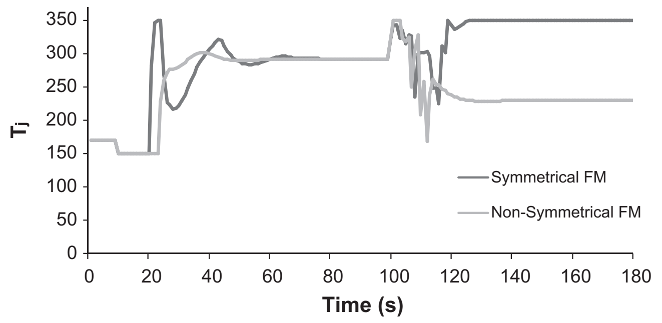

Fig. 4c. Case Study II, first case: Control moves for the two MPC schemes.

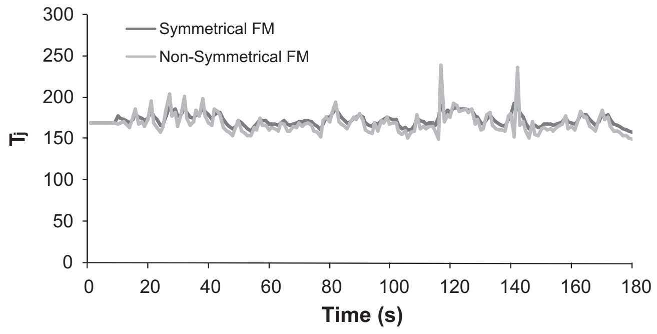

Fig. 5c. Case Study II, second case: Control moves for the two MPC schemes.

then applied to two test cases. In the first case, the objective is to drive the CSTR through the three steady state points, thus testing whether the controllers are capable of controlling the reactor throughout its operating range. The CSTR is initiated at a concentration C_A equal to 0.2 which corresponds to the lower steady state point and then a step change in the set-point from 0.2 to 0.95, i.e. the upper stead state point is introduced. After the CSTR reaches the new set point, at time point 100 a new step change is applied, leading to the middle unstable steady state point (0.6). The responses are depicted in Fig. 4a while Fig. 4b presents the computational times needed by the two schemes to solve the optimization problem at each discrete time instance and Fig. 4c gives the respective control moves. The dynamic response of the controller using the non-symmetric algorithm while attempting to track the first set point change is slightly better as it reaches faster the new set point with fewer oscillations. The superiority of the scheme based on the non-symmetric algorithm becomes more apparent during the second set point change towards the unstable steady state point. The controller using the non-symmetric algorithm manages to reach the set point, taking advantage of the more accurate model it uses for prediction, while the controller using the symmetric algorithm totally misses the set point, as it gets stuck to the lower stable steady state point. The failure of the symmetric controller to reach the set point can be explained by noticing that the initial modeling error for the two state variables is propagated throughout the prediction horizon. The initial error is amplified because the model predictions at each time step are used for calculating the predictions in the subsequent time instance.

Another important observation is that the time needed to solve the optimization problem in the non-symmetric MPC scheme is considerably lower compared to the time needed for the symmetric scheme, which is obviously induced by the reduction in the size of networks generated by the non-symmetric algorithm. The reduction in computational time is more significant at the time instances that follow the set point changes. It should be noted that at these particular time instances the optimization problems are harder to solve, because there are considerable differences between the optimal solutions at consecutive time steps and thus no good initial guesses are available.

Table 6
Sum of Squared Error (SSE) for the two algorithms during the disturbance rejection test.

	MPC with symmetric FM	MPC with non-symmetric FM
SSE	0.1343	0.0817

In the second case the set point was fixed at 0.20 and unknown disturbances were introduced to the inlet flow *F*. The disturbances were assumed to follow a normal distribution with mean equal to 20 and standard deviation equal to 3.5. The results are summarized in Figs. 5a–5c where the responses of the two control schemes, the associated computational times and the control moves are depicted, and in Table 6, showing the Sum of Squared Errors (SSE) for the two controllers. Once again, the MPC scheme based on the non-symmetric fuzzy means algorithm proves more accurate, featuring an SSE that is 39% lower compared to the controller based on the symmetric algorithm, while at the same time it proves to be considerably faster in solving the optimization problem.

6. Conclusions

This paper presents a new algorithm that addresses the problem of calculating the number and locations of the hidden node centers, in the process of training an RBF network. The new algorithm relies on a fuzzy partition of the input space where each input variable is allowed to be partitioned into a different number of fuzzy sets. This results to a non-symmetric fuzzy partition of the input space. A hyper-ellipsoid equation is employed to define a new multidimensional membership function to a fuzzy subspace. The new algorithm improves the efficiency of the original fuzzy means algorithm in terms of accuracy, but also produces networks of lower complexity. The application of the proposed methodology in developing dynamic RBF models for a benchmark modeling problem, illustrated the efficiency of the new method. The advantages of the method to produce more accurate and smaller in size RBF networks are highly appreciated when the RBF model is integrated in online control applications where both model accuracy and computational time are critical issues. In order to demonstrate this, a model produced by the non-symmetric fuzzy means algorithm was incorporated in an MPC framework and was compared with a model produced by the symmetric fuzzy means algorithm. The proposed approach not only improves considerably the performance of the controller, but also solves the on-line optimization problem in considerably lower computational times.

Acknowledgment

Alex Alexandridis gratefully acknowledges financial support from the Greek State Scholarships Foundation.

References

[1] Alexandridis A, Sarimveis H, Bafas G. A new algorithm for online structure and parameter adaptation of RBF networks. Neural Networks 2003;16:1003–17.
[2] Alexandridis A, Sarimveis H. Nonlinear adaptive model predictive control based on self-correcting neural network models. AICHE J 2005;51(9):2495–506.
[3] Bhartiya S, Whiteley JR. Factorized approach to nonlinear MPC using a radial basis function model. AIChE J 2001;47:358–68.
[4] Bhartiya S, Whiteley JR. Benefits of factorized RBF-based NMPC. Comput Chem Eng 2002;26:1185–99.
[5] Darken C, Moody J. Fast adaptive k-means clustering: some empirical results. In: IEEE INNS international joint conference on neural networks; proceedings, vol. 2; 1990. p. 233–8.
[6] Hefny HA, Bahnasawi AA, Abdel Wahab AH, Shaheen SI. Logical radial basis function networks a hybrid intelligent model for function approximation. Adv Eng Softw 1999;30:407–17.
[7] Hu Z, Chan CW, Huang GH. Model predictive control for in situ bioremediation system. Adv Eng Softw 2006;37(8):514–21.
[8] Jiang N, Zhao Z, Ren L. Design of structural modular neural networks with genetic algorithm. Adv Eng Softw 2003;34(1):17–24.
[9] Kazantzis N, Kravaris C. Synthesis of state feedback regulators for nonlinear processes. Chem Eng Sci 2000;55:3437.
[10] Lazaro M, Santamarıa M, Pantaleon C. A new EM-based training algorithm for RBF networks. Neural Networks 2003;16:69–77.
[11] Mackey MC, Glass L. Oscillation and chaos in physiological control systems. Science 1977;197:287–9.
[12] MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, UC Berkeley Press; 1967. p. 281–97.
[13] Nie J. Fuzzy control of multivariable nonlinear servomechanisms with explicit decoupling scheme. IEEE Trans Fuzzy Syst 1997;5:304.
[14] Panchapakesan C, Palaniswami M, Ralph D, Manzie C. Effects of moving the centers in an RBF network. IEEE Trans Neural Networks 2002;13(6):1299–307.
[15] Peng H, Ozaki T, Haggan-Ozaki V, Toyoda Y. A parameter optimization method for radial basis function type models. IEEE Trans Neural Networks 2003;14(2):432–8.
[16] Peng H, Nakano K, Shioya H. Nonlinear predictive control using neural nets-based local linearization ARX model – stability and industrial application. IEEE Trans Control Syst Technol 2007;15(1):130–43.
[17] Peng JX, Li K, Huang DS. A hybrid forward algorithm for RBF neural network construction. IEEE Trans Neural Networks 2006;17(6):1439–51.
[18] Sarimveis H, Alexandridis A, Tsekouras G, Bafas G. 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 2002;41:751–9.
[19] Sarimveis H, Alexandridis A, Mazarakis S, Bafas G. A new algorithm for developing dynamic radial basis function neural network models based on genetic algorithms. Comput Chem Eng 2004;28(1–2):209–17.
[20] Zhao ZQ, Huang DS. A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability. Appl Math Modell 2007;31:1271–81.
[21] Wang SW, Yu DL, Gomm JB, Page GF, Douglas SS. Adaptive neural network model based predictive control for air–fuel ratio of SI engines. Eng Appl Artif Intell 2006;19:189–200.
[22] Wedge D, Ingram D, McLean D, Mingham C, Bandar Z. On global–local artificial neural networks for function approximation. IEEE Trans Neural Networks 2006;17(4):942–52.