System Identification through RBF Neural Networks: Improving Accuracy by a Numerical Approximation Method for the Centroids and Widths Adjustment

Paulo D. L. de Oliveira[1], Arthur P. de S. Braga[1], Laurinda L. N. dos Reis[1], Fabrício G. Nogueira[1], Antônio B. de S. Júnior[1].

[1] Federal University of Ceará, Fortaleza, Brazil
pdaving@gmail.com;
{arthurp, laurinda, fnogueira, barbosa}@dee.ufc.br

Abstract. Within the last two decades there has been an increasing need for the development of mathematical models out of observed data captured from a system, a process called empirical modelling or systems identification. Under this circumstance, many techniques and methodologies have been proposed, among them the use of Artificial Neural Networks. It is proposed herein a non-hybrid gradient-based learning algorithm for a Radial Basis Function Neural Network aimed at improving the accuracy of non-linear dynamical system modelling. A single-stage non-hybrid approach is employed for the learning process, where the free parameters of the network – the centroids positioning, the receptive fields width, and the weights – are updated through a supervised method. Accurate identification capability is examined by the use of two non-linear datasets and the performance of the proposed method is compared with traditional techniques. Results demonstrate that nonlinear system identification can be significantly improved with easy-to-implement gradient-based RBF learning strategy.

Keywords: Artificial Neural Networks; System Identification; RBF; Parameter Estimation; Gradient Descent; Supervised Learning.

1 Introduction

Systems Identification is a research field that deals with the determination of mathematical models of a system for particular purposes, such as prediction, control, diagnosis etc. [1].

Either because of the increasing complexity of the systems or for the availability of computational power and tools for capturing data, obtaining the mathematical model of a dynamic system is going beyond the often-complex process of deriving its properties from physical laws. In light of that, the method known as *Black Box System Identification* [1] becomes very useful when: 1) it is prohibitive to raise the equations regarding the nature of the process under modelling, and 2) a set of data can be collected from the system's behavior when it is under operation. It is noteworthy that few – if none –

previous assumptions about the nature of the system is required when using the black box modelling strategy. The key information is the collection of input and output data from which a relation can be inferred through any technique known in the area of systems identification.

One of the techniques extensively employed for this task is the use of Artificial Neural Networks (ANNs) [1, 9], especially regarding nonlinear dynamical systems. Among numerous classes of networks, the Radial Basis Function (RBF) neural networks have attracted much attention, probably due to its simpler structure, faster learning capability and easy-to-implement algorithms. The key to its superior performance is intrinsically related to the parameter adjustments, mainly the centroid positioning, the width of the adopted basis function and the weights between the hidden and output layers. Typical methodologies for training such networks often consider a hybrid approach for the parameter estimation and the learning process, i.e., an unsupervised technique followed by a supervised one [2-4, 6-8, 10-12, 14-17]. This work, in contrast, proposes a single-stage learning algorithm that simultaneously updates the RBF parameters through a supervised gradient-based approach.

In the recent literature, many studies have dealt with multi-stage learning algorithms [6, 10, 12]; clustering algorithms for centroid relocation [4, 6-8, 10, 12, 13]; and optimal width adjustment for the basis functions [2, 3, 6, 11, 12, 14, 17]. To briefly describe a few, a three-stage algorithm, comprising *k*-means, *k*-nn and Optimal Steepest Descent (OSD) methods, is proposed in [12]. Kayhan et al. [10] also propose a three-stage approach in which the RBF parameters are tuned in a two-stage pre-process and then the gradient descent algorithm is used for updating network weights. In [6] the authors modify their previous work [12] for improving the center relocation strategy by using Particle Swarm Optimization (PSO), followed by a *k*-nn and OSD algorithms for width adjustment and weight updating, respectively.

In [17] special attention to optimizing the widths is taken, along with a method for selecting the appropriate number of basis functions. For determining the centroid positions, the widths of the basis functions and the weights of the network, a *k*-means clustering along with an improved Orthogonal Least Squares (OLS) algorithms are examined in [4]. Kitayama et al. [11] propose a simple width estimate which is based on a well-known equation given by Haykin [9]. Their proposal introduces an adaptive scaling coefficient for improving Haykin's fixed estimate. Based on the concepts of Renyi's Entropy, the widths of the gaussian functions are continuously adapted in [14]. In [15], the authors propose a new clustering algorithm that uses the network's input and output information for better relocating the centroids and for determining an optimal number of basis functions. An iterative approach is used for adjusting widths and weights in [16], while the centers are uniformly distributed in a previous stage of the proposed algorithm.

In [13] a complete supervised technique is used for continuously updating the weights and selecting, rather than updating, the center positions. The work herein presented proposes similar strategy but does not require excessive – and incremental – memory resources. It instead updates, iteratively, the centroid positions and the receptive fields by a gradient descent algorithm, just like it is traditionally done to the network weights. Additionally, a second variant of the algorithm performs the adjustments

to the centroid positions and network weights, while using conventional technique for the widths. The performance of such approaches will be examined and validated in section 4.

The next section will briefly discuss the fundamentals of Radial Basis Function neural networks, while section 3 details about the proposed algorithms for training such networks. The outcomes will be presented in section 4, followed by the author's conclusions of the work herein presented.

2 RBF Neural Networks

Beyond the stablished capability for solving pattern classification problems, Radial Basis Function Neural Networks (RBFNNs) have been widely used for function approximation purposes, such as in systems identification. RBFNNs are typically structured with only one hidden layer (see Figure 1), with m neurons, where the activation functions are often defined to be Gaussian, as in Equation 1:

$$\Phi_j(\mathbf{x}, \mathbf{c}_j, \sigma_j) = e^{\frac{-\|\mathbf{x}-\mathbf{c}_j\|^2}{\sigma_j^2}} \; ; \; j = 1, \ldots, m \tag{1}$$

where the vector $\mathbf{c}_j$ and the scalar σ_j are the parameters of the gaussian function referred to as the centroid and the width, respectively. The vector $\mathbf{x}$, with dimensionality D, represents a sample of the input pattern set that enters the neural network.

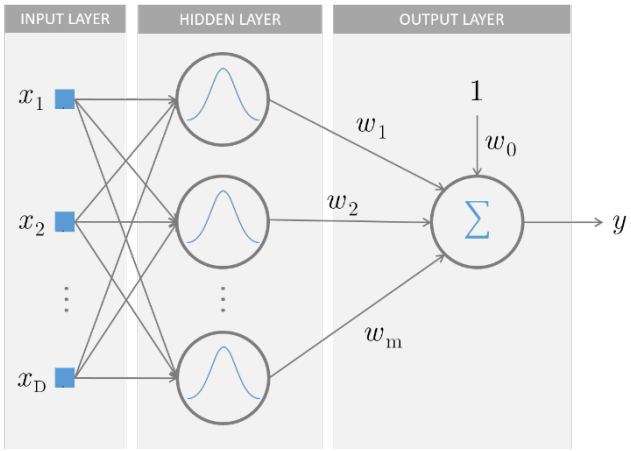

Fig. 1. Typical structure of an RBF Neural Network.

At the hidden layer of the RBF neural network, the distance – the Euclidean distance, for instance – from any sample of the input data to the centroids of the gaussians defines the degree of activation of any neuron. It is, thereby, reasonably expected that the centers are located within the input space so that proper activation may take place. This property is directly related to the accuracy and the ability of the network to learn from data. Another key characteristic of RBF neural networks is the receptive field (width)

of the basis functions. The effects of changing the widths may lead the network, in extreme cases, to over-smooth or to over-fit the function to which approximation is intended. It is, hence, intuitive that proper determination of these parameters will increase the ability and accuracy of an RBFNN to approximate a function.

Training an RBFNN typically involves two stages [5], one for tuning the parameters of the gaussian functions in the hidden layer; and the other one refers to the adjustment of the weights of the network to fit the desired output data as close as possible. This is known as hybrid learning process, in which the first stage is related to an unsupervised learning – such as clustering algorithms – and the second stage employs a supervised process, typically Least Squares, to mention one.

Another possibility for training an RBFNN is related to a complete supervised process [13], both to optimally adjust the network parameters and to simultaneously update weights in order to fit the desired data accordingly. This non-hybrid approach is employed in this work, where the gradient descent method is used for optimal parameter estimation. Section 3 will explain in detail the learning strategy used for the RBF architecture herein presented.

3 Non-hybrid Learning for RBFNNs

Following a complete non-hybrid approach, the authors first developed an algorithm for centroids, widths and weights adjustments based on the same technique: The Gradient Descent (GD). In such algorithm, partial derivatives of a cost function with respect to all three parameters must be iteratively estimated during the learning process.

In a second approach, only centroids and weights are adjusted following the gradient descent method. Receptive fields are iteratively updated through Eq. 2:

$$\sigma(i) = \frac{d_{max}(i)}{\sqrt{m}} \tag{2}$$

where m is the number of neurons in the hidden layer and $d_{max}(i)$ is given by Eq. 3:

$$d_{max}(i) = \max_{1 \le j \le m} \left[\max_{1 \le k \le m} \left\| \mathrm{c}_j(i) - \mathrm{c}_k(i) \right\| \right] \tag{3}$$

in which $c_j(i)$ and $c_k(i)$ represent the centroid vectors for every neuron in the hidden layer at iteration $i = 1, \dots, N$. It is important to mention that the dimensionality of the centroids must match the dimensionality D of the input samples, as stated in section 2.

3.1 First Approach: Complete Supervised Learning

This approach will be referred to as AP#1. Let N be the number of input and output samples available for training the network. Assuming a cost function as:

$$E(i) = \frac{\left(y(i) - \hat{y}(i)\right)^2}{2}; \quad i = 1, \dots, N \tag{4}$$

where $y(i)$ and $\hat{y}(i)$ are the desired output and the estimated output of the network, respectively, at iteration i of the algorithm at any epoch. The learning process must calculate the partial derivatives of (4) with respect to the centroids ($\boldsymbol{c}$), the widths (σ) and the weights (ω), as in the following equations:

$$\frac{\partial E(i)}{\partial c_{j,k}(i)} = -\big(y(i) - \hat{y}(i)\big)\frac{\partial \hat{y}(i)}{\partial c_{j,k}(i)}; \quad i = 1, \dots, N; \;\; j = 1, \dots, m; \;\; k = 1, \dots, D \tag{5}$$

$$\frac{\partial E(i)}{\partial \sigma_j(i)} = -\big(y(i) - \hat{y}(i)\big)\frac{\partial \hat{y}(i)}{\partial \sigma_j(i)}; \quad i = 1, \dots, N; \;\; j = 1, \dots, m; \tag{6}$$

$$\frac{\partial E(i)}{\partial \omega_j(i)} = -\big(y(i) - \hat{y}(i)\big)\frac{\partial \hat{y}(i)}{\partial \omega_j(i)}; \quad i = 1, \dots, N; \;\; j = 1, \dots, m; \tag{7}$$

In equations (5) and (6), the partial derivatives of the estimated output with respect to the centroids and widths can be worked out by numerical approximations. In equation (7), the partial derivatives are:

$$\frac{\partial \hat{y}(i)}{\partial \omega_j(i)} = \Phi\big(\mathbf{x}_i, \boldsymbol{c}_j, \sigma_j\big); \quad i = 1, \dots, N; \;\; j = 1, \dots, m. \tag{8}$$

$$\frac{\partial \hat{y}(i)}{\partial \omega_0(i)} = 1; \quad i = 1, \dots, N; \tag{9}$$

Hence, the gradient descent updating equations for these parameters should be:

$$c_{j,k}(i+1) = c_{j,k}(i) + \alpha\big(y(i) - \hat{y}(i)\big)\frac{\partial \hat{y}(i)}{\partial c_{j,k}(i)}; \tag{10}$$

$$i = 1, \dots, N; \; j = 1, \dots, m; \; k = 1, \dots, D$$

$$\sigma_j(i+1) = \sigma_j(i) + \alpha\big(y(i) - \hat{y}(i)\big)\frac{\partial \hat{y}(i)}{\partial \sigma_j(i)}; \tag{11}$$

$$i = 1, \dots, N; \;\; j = 1, \dots, m;$$

$$\omega_j(i+1) = \omega_j(i) + \alpha\big(y(i) - \hat{y}(i)\big)\frac{\partial \hat{y}(i)}{\partial \omega_j(i)}; \tag{12}$$

$$i = 1, \dots, N; \;\; j = 1, \dots, m;$$

$$\omega_0(i+1) = \omega_0(i) + \alpha\big(y(i) - \hat{y}(i)\big); \tag{13}$$

$$i = 1, \dots, N;$$

From (10) to (13), the parameter α is the learning rate of the network and is directly related to the speed of convergence of the learning process. The term ω_0 in Eqs. (9) and (13) refers to the bias of the network – a weight whose input is a constant value, often set to 1 (see Fig.1).

These steps are performed for every new input sample. After *N* samples have been provided (one epoch elapsed) the algorithm runs again until a stop criterion has been met.

3.2 Second Approach: Supervised Adjustment for Centroids and Weights

In this approach, referred to as AP#2, the rules for deriving the updating equations for the centroids and weights are the same as for the first approach (AP#1), so there is no need to restate the equations. The only difference is in the update of the receptive fields, which are calculated by Eqs. (2) and (3) at each iteration $i = 1, \ldots, N$.

As for the first approach, the algorithm runs repeatedly until a stop criterion has been met after each epoch.

4 Simulation Results

In this section, both versions (AP#1 and AP#2) of the learning algorithm are tested and validated using two datasets regarding real data captured from non-linear processes.

The system can be considered as identified when the estimated outputs match the desired values to a certain extent based on a similarity measure. MATLAB® provides a useful formula to determine how well one signal matches a reference. This goodness of fit is calculated as:

$$\text{fit} = 1 - \frac{\|\mathbf{y} - \hat{\mathbf{y}}\|}{\|\mathbf{y} - mean(\mathbf{y})\|} \tag{14}$$

which indicates that the closer fit is to 1, the more vector $\hat{\mathbf{y}}$ is similar to $\mathbf{y}$. Negative values indicate a poor similarity. This measurement is used throughout this section and reflects the accuracy of the approximation achieved in different simulations.

4.1 Nonlinear Water Level Tank

The first process refers to a water tank SISO (Single-Input Single-Output) system with half-conical half-cylindrical shape. In this dynamic system, the input is the water flow, and the output is the water level inside the tank. A total of 10000 samples is available, from which 8000 are dedicated for the training stage, while the remaining 2000 are used for validation. Figure 2 depicts a window of 600 samples of the input data, along with the target output.

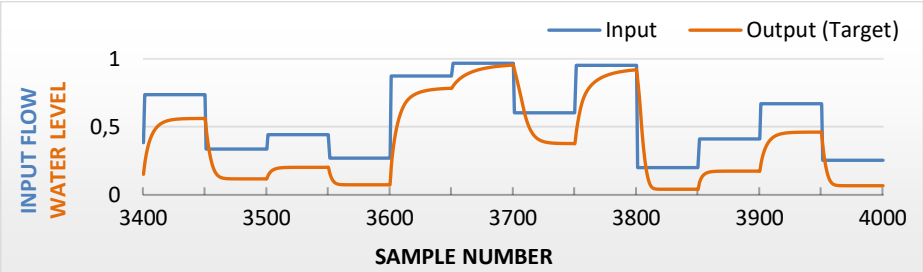

Fig. 2. 600 (out of 10000) Input and Output samples of the nonlinear water tank system. Both signals are normalized.

The input and output signals are normalized before applied to the RBFNN input layer. The matrix of regressors is composed by the input signal and delayed outputs fed back into the network in the form: $\mathbf{x}(i) = [\, u(i) \;\; y(i-1) \,]$; $i = 1, \ldots, N$. This represents an ARX structure in which the current output $\hat{y}(i)$ is derived from the past output $y(i-1)$ and the current input $u(i)$.

4.2 Nonlinear pH Neutralization Process

The second dataset refers to a MISO (Multiple-Input Single-Output) chemical process of pH neutralization in a stirring tank, in which the flow of an acid and a base solution are the inputs of the process, while the output is the pH of the mixed solution. This process exhibits high non-linear properties and provides a stricter baseline for evaluating the capability of an RBFNN in identifying nonlinear models.

The input and output signals are also normalized before applied to the RBFNN input layer. The matrix of regressors is in the form: $\mathbf{x}(i) = [\, u_1(i) \;\; u_2(i) \;\; y(i-1) \,]$; $i = 1, \ldots, N$, where $u_1(i)$ is the input flow of the acid solution; $u_2(i)$ the input flow of the base solution; and $y(i-1)$ is the past output pH of the mixed solution.

A total of 2000 samples is available, from which 1600 (80%) and 400 (20%) samples are used as the training and validation data, respectively. Figures 3 and 4 present a window of 50 samples of both inputs and the target output, respectively.

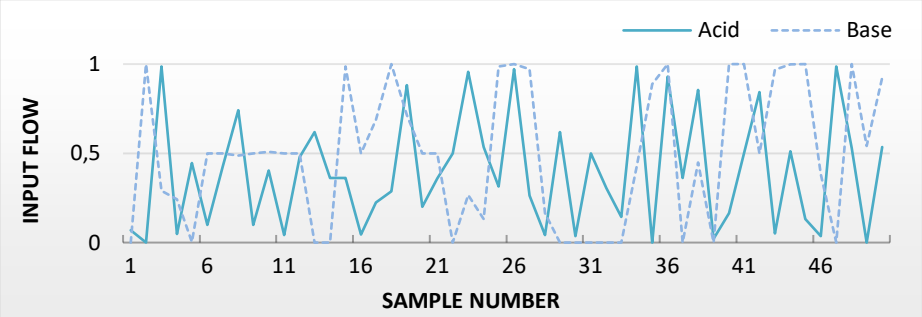

Fig. 3. First 50 (out of 2000) input samples of the high nonlinear chemical process. Both inputs are normalized.

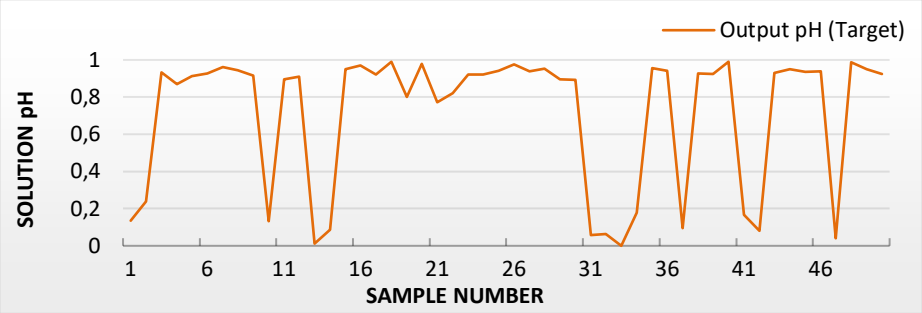

Fig. 4. First 50 samples (out of 2000) of the normalized output of the high nonlinear chemical process.

4.3 Methodology

It is noticeable that depending on the initialization of the free parameters, the RBFNN may provide different results. To counteract this randomness, every simulation has been run 20 times, with different initialization values for the weights, centroids and widths of the network at each run. The graphics in Figures 5 and 6 are plotted by taking the mean value of the 20 simulations for every number of hidden units.

For all datasets, which have been split into 80% of training and 20% of validation data, both versions of the proposed algorithm (AP#1 and AP#2) are compared to a classical hybrid approach that employs the *k*-means clustering in the unsupervised stage plus a gradient-descent method in the supervised one. This first reference algorithm will be referred to as *k*-means. Additionally, the use of MATLAB® System Identification Toolbox is also presented as a second reference, which will be referred to as IDENT.

The *k*-means algorithm intends to optimally relocate the RBF centroids within the input data space by iteratively updating every centroid position according to their current distance, e.g. the Euclidean distance, from the current input sample [5, 9]. This, however, does not relate the centroids positioning to the approximation error of the RBFNN whose minimization is intended.

As for the System Identification toolbox, it employs the traditional Least Squares method [18] whose aim is to minimize the sum of the squares of the residuals in the estimates. For polynomial model estimates, previous assumption of the system order is required. For this, the toolbox allows manual or automated specification. The authors tested both strategies and picked the one that achieved the best results, namely, the ARX 130 (na=1, nb=3, nk=0) for the water tank system and ARX 140 for the chemical process. This means that the difference equation for the tank system is:

$$y(t) = -a_1 y(t-1) + b_1 u(t) + b_2 u(t-1) + b_3 u(t-2) + e(t) \qquad (15)$$

And the difference equation for the chemical process is:

$$y(t) = -a_1 y(t-1) + b_1 u(t) + b_2 u(t-1) + b_3 u(t-2) + \mathrm{b}_4 u(t-3) + e(t) \qquad (16)$$

Fig. 5 brings the results of the identification, in terms of the similarity measure in Eq.14, of the tank system for a varying number of centroids. It can be seen that the proposed algorithm (blue/orange lines) outperforms *k*-means for any number of hidden units (same as centroids), both for the training and validation data. It also performs better than IDENT, although it requires more than seven hidden units in general.

Fig. 6 shows the results for the chemical process. It is seen that for few hidden units, the RBFNN fittings (orange and blue curves) are similar to those obtained by IDENT. However, it also outperforms this reference method as the number of hidden neurons grows. It is also noticeable that the proposed algorithm, in general, achieves better fitting when compared to *k*-means for this high nonlinear system.

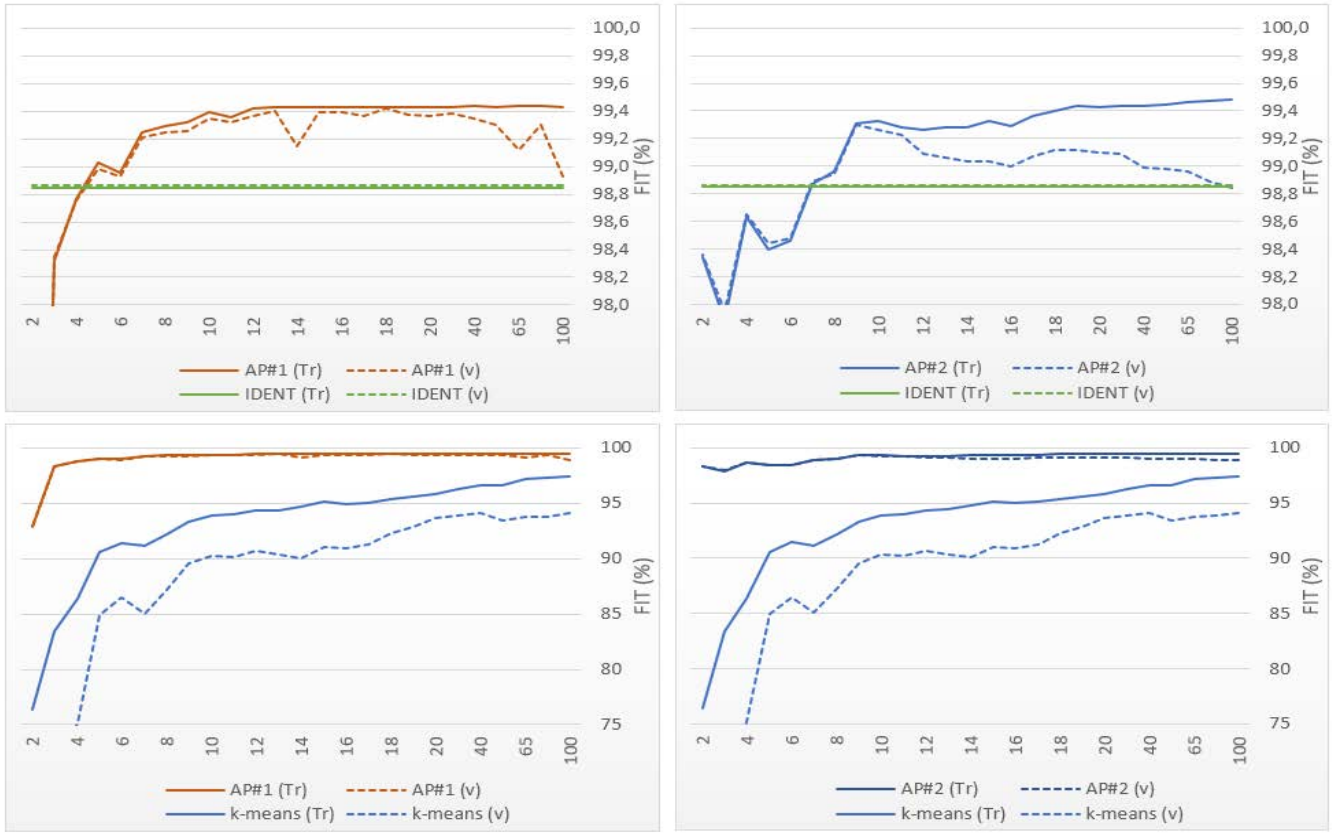

Fig. 5. Simulation results for the tank process for both versions of the proposed algorithm compared to IDENT and *k*-means. Horizontal axis refers to the number of hidden units for AP#1, AP#2 and *k*-means. (Tr) means Training data and (v) means Validation data.

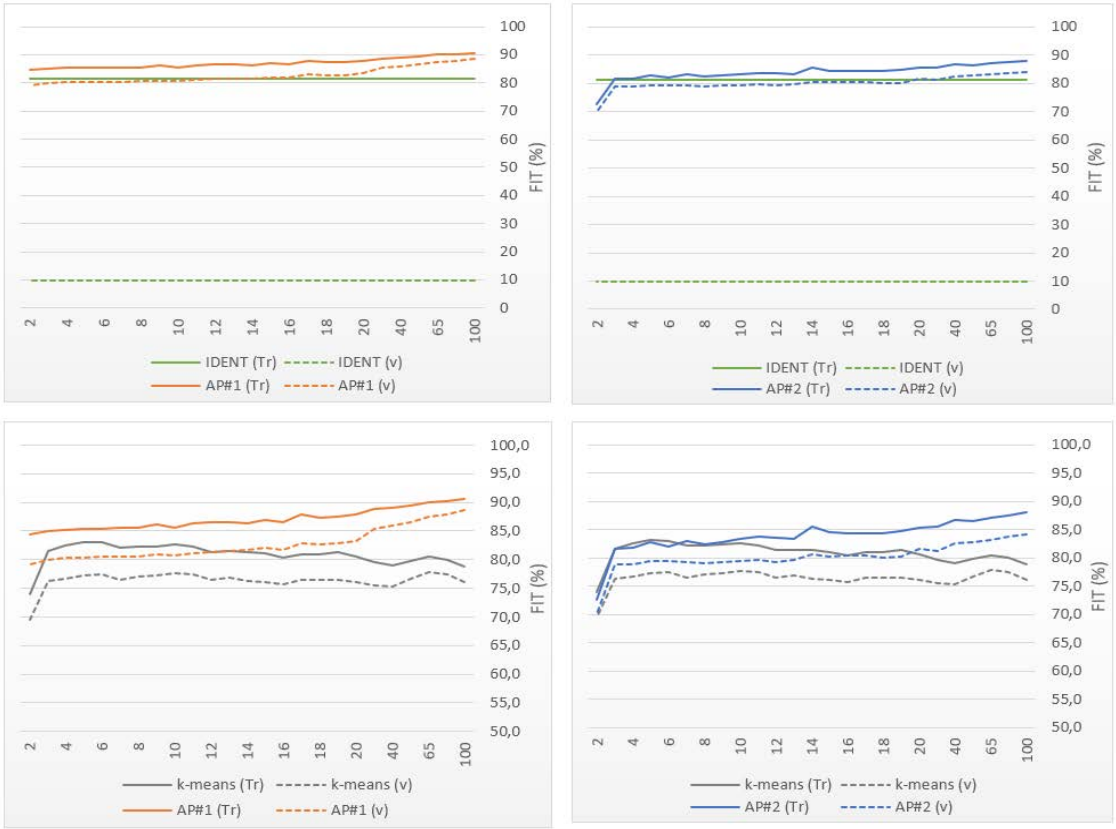

Fig. 6. Simulation results for the chemical process for both versions of the proposed algorithm compared to IDENT and *k*-means. Horizontal axis refers to the number of hidden units for AP#1, AP#2 and *k*-means. (Tr) means Training data and (v) means Validation data.

5 Concluding Remarks

Radial Basis Function Neural Networks with one-stage supervised learning process have shown to provide excellent accuracy on determining models for non-liner systems. By achieving superior accuracy on the estimates, the authors understand that tuning all the RBFNN parameters with gradient-based techniques are of great importance for function approximation tasks that demand high quality models.

The choice for the two datasets regarding real nonlinear systems was made to provide different perspectives based on the degree of nonlinearity that each system exhibits. The dynamics of the tank system can be easily captured with traditional techniques such as Least Squares and neural networks using *k*-means. For such system, not surprisingly, the RBFNN learning algorithm proposed in this work performed very well, although there is a slight trend of loss in the generalization of the network learning, as the curves for validation data (dashed lines in Fig.5) are decreasing.

To introduce a degree of difficulty in the validation of the RBFNN technique for system identification purposes, the high nonlinear chemical process has also been included which demonstrated the capability of such technique in capturing different levels of nonlinearities with considerable accuracy.

For future works, the authors may treat criteria for stop tuning the free parameters of the network as well as the ability of the RBFNN to self-organize, i.e., grow and/or decrease its structure according to some criteria, as in [7, 8]. Those capabilities are supposed to achieve better generalization for a greater number of nonlinear systems.

Acknowledgments

The authors wish to thank the Brazilian National Council for Scientific and Technological Development (CNPq) for its universal call for project 442573/2014-6 through which the development and publication of this work has been possible. The authors should also thank the Coordination for the Improvement of Higher Education Personnel (CAPES) for the financial support; and the Research Group of Automation and Robotics (GPAR/UFC) for the laboratory infrastructure made available.

References

1. L.A. Aguirre. “Introdução à Identificação de Sistemas: Técnicas Lineares e Não-Lineares aplicadas a sistemas reais”. UFMG Press. (2000).
2. N. Benoudjit, C. Archambeau, A. Lendasse, J. Lee, M. Verleysen, “Width optimization of the Gaussian kernels in Radial basis function networks,” ESANN’2002, pp. 425–432, (2002).
3. N. Benoudjit, M. Verleysen, “On the kernel widths in radial-basis function networks,” Neural Process. Lett., vol. 18, no. 2, pp. 139–154, (2003).
4. S. Billings, H.-L. Wei, M. Balikhin, “Generalized multiscale radial basis function networks.,” Neural Netw., vol. 20, no. 10, pp. 1081–1094, (2007).
5. A. P. de S. Braga. “Redes Neurais Artificiais”. Lecture Notes. (2017).

6. V. Fathi, G. A. Montazer, “An improvement in RBF learning algorithm based on PSO for real time applications,” Neurocomputing, vol. 111, pp. 169–176, (2013).
7. H. Han, Q. L. Chen, J. Qiao, “An efficient self-organizing RBF neural network for water quality prediction.,” Neural Netw., vol. 24, no. 7, pp. 717–725, (2011).
8. H. Han, W. Zhou, J. Qiao, G. Feng, “A Direct Self-Constructing Neural Controller Design for a Class of Nonlinear Systems,” vol. 26, no. 6, pp. 1312–1322, (2015).
9. S. Haykin. “Neural Networks - A Comprehensive Foundation (2nd ed.)”, Prentice Hall (1999).
10. G. Kayhan, A. E. Ozdemir, İ. Eminoglu, “Reviewing and designing pre-processing units for RBF networks: initial structure identification and coarse-tuning of free parameters,” Neural Comput. Appl., vol. 22, no. 7–8, pp. 1655–1666, (2013).
11. S. Kitayama, K. Yamazaki, “Simple estimate of the width in Gaussian kernel with adaptive scaling technique,” Appl. Soft Comput. J., vol. 11, no. 8, pp. 4726–4737, (2011).
12. G. A. Montazer, R. Sabzevari, F. Ghorbani, “Three-phase strategy for the OSD learning method in RBF neural networks,” Neurocomputing, vol. 72, no. 7–9, pp. 1797–1802, (2009).
13. K. Okamoto, S. Ozawa, S. Abe, “A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory,” Int. Jt. Conf. Neural Networks, (2003).
14. A. Singh, J. C. Príncipe, “Information theoretic learning with adaptive kernels,” Signal Processing, vol. 91, no. 2, pp. 203–213, (2011).
15. D. Wang, X.-J. Zeng, J. A. Keane, “A clustering algorithm for radial basis function neural network initialization,” Neurocomputing, vol. 77, no. 1, pp. 144–155, (2012).
16. P. Xu, A. W. Jayawardena, W. K. Li, “Model selection for RBF network via generalized degree of freedom,” Neurocomputing, vol. 99, pp. 163–171, (2013).
17. P. Zhou, D. Li, H. Wu, F. Cheng, “The automatic model selection and variable kernel width for RBF neural networks,” Neurocomputing, vol. 74, no. 17, pp. 3628–3637, (2011).
18. L. Ljung. “System Identification Toolbox User’s Guide”, Version 5, MathWorks Press, Natick MA, p.201, (2000).