ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **12**, Number 1, 2009, 69–90

Boosting the Performances of the Recurrent Neural Network by the Fuzzy Min-Max

Ryad ZEMOURI[1], Florin Gheorghe FILIP[2], Eugenia MINCA[3], Daniel RACOCEANU[4], Noureddine ZERHOUNI[5]

[1] Laboratoire d'automatique, CNAM,
21 rue Pinel, 75013 Paris, France
E-mail: ryad.zemouri@cnam.fr

[2] Romanian Academy-INCE and ICI,
Calea Victoriei 125, Bucharest, Romania
E-mail: ffilip@acad.ro

[3] Faculty of Electric Engineering, Valahia University,
Bd. Unirii, nr. 18, 0200, Târgovişte, Romania
E-mail: minca@valahia.ro

[4] IPAL (UMI CNRS 2955, UJF, I2R/A*STAR, NUS),
1 Fusionopolis Way, #21-01 Connexis, 138632 Singapore
E-mail: danielr@comp.nus.edu.sg

[5] FEMTO-ST - UMR CNRS 6174, ENSMM, UFC, UTBM,
32 Avenue de l'Observatoire, 25044 Besançon, France
E-mail: zerhouni@ens2m.fr

Abstract. The k-means training algorithm used for the RBF (Radial Basis Function) neural network can have some weakness like empty clusters, the choice of the cluster number and the random choice of the centers of theses clusters. In this paper, we use the Fuzzy Min Max technique to boost the performances of the training algorithm. This technique is used to determine the number of the k centers and to initialize correctly these k centers. The k-means algorithm always converges to the same result for all the tests.

Key words: Fuzzy Min-Max, k-means algorithm, RBF network, recurrent network, time series prediction.

1. Introduction

Artificial Neural Networks (ANNs) have emerged as well-known solutions for tackling pattern recognition and classification tasks. In industrial fault detection and classification applications, extensive studies using ANNs to detect faults and to identify the associated causes have been conducted [1], [2], [3]. ANNs have been also utilized in decision-making processes to identify the type of the decision problem with the view of choosing the adequate model for analysis and solving [4, 5]. To consider the temporal aspect of the input data, ANNs require some modifications of the static neural models [6]. Two different approaches of the time representation in the neural network architectures can be identified [7]. In the first case, the time is represented as an external mechanism. Elman [8] names this technique "spatial representation of time". In the second case, the neural network is able to treat the time dimension without any external mechanism. The *Recurrent Neural Networks* (RNN)are fundamentally different from *feed-forward architectures* in the sense that they not only operate on an input space but also on an internal *state* space – a trace of what already has been processed by the network. This is equivalent to an *Iterated Function System* (IFS; see [9] for a general introduction to IFSs; [10] for a neural network perspective) or a *Dynamical System* (DS; see *e.g.* [11, 12] for a general introduction to dynamical systems; ([13]; [14]) for neural network perspectives).

In some particular situations, fault prediction applications can be viewed as time series prediction processes. A significant amount of work has been done in the realm of time series analysis, modeling, and prediction to support analysis and interpretation of such data [15]. The commonly encountered models of time series include autoregressive models [16], recurrent neural networks [17], [18], [19], [20] and fuzzy rule-based models [17], [21], [22], [23].

The *Recurrent Radial Basis Functions* (RRBF) networks were introduced in [24],[2]. The dynamic aspect is obtained by the use of an additional self-connection on the input neurons with a sigmoid activation function. The RRBF network can thus take into account a certain past of the input signal (Fig. 1). The network parameters are determined with a two stage training process. During the first stage, an unsupervised k-means learning algorithm is used to determine the parameters of the RBF nodes (the centers and the influence rays of the Gaussian nodes). In the second stage, linear regression is used to determine the weights between the hidden and the output layer.

In the case of linear output nodes, the network output is expressed as follows [25]:

$$y_k = \sum_{j=1}^{M} w_{kj}\phi_j(\|\mathbf{x} - \mathbf{u}_j\|) + w_{k0}, \tag{1}$$

where $\mathbf{x}$ is the input vector with elements x_i (where i is the dimension of the input vector); $\mathbf{u}_j$ is the vector determining the centre of the basis function ϕ_j with elements u_{ji}; w_{kj} are the final layer weights and w_{k0} is the bias. The basis function $\phi_j(.)$ provides the non-linearity. Each neuron of the input layer gives a summation at the instant t between its input x_i and its previous output weighted by a self-connection

w_{ii}: The output of its activation function is:

$$\begin{aligned} a_i(t) &= w_{ii}\xi_i(t-1) + x_i(t), \\ \xi_i(t) &= f\left(a_i(t)\right), \end{aligned} \tag{2}$$

where $a_i(t)$ and $\xi_i(t)$ represent respectively the neuron activation and its output at the instant t, f is the sigmoid activation function:

$$f\left(x\right) = \frac{1-\exp(-kx)}{1+\exp(-kx)}. \tag{3}$$

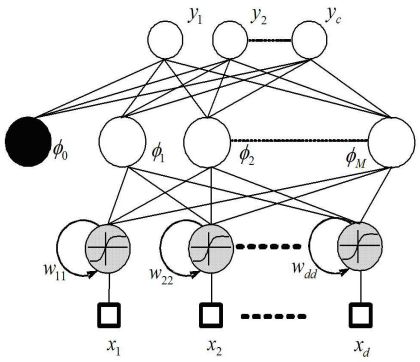


Fig. 1. Recurrent Radial Basis Functions networks.

By making some tests with the classic version of the k-means algorithm, we note that they have three weaknesses as it follows:

1. There might be some situations where a points' cluster can be empty. This generates a problem to calculate the prototype influence ray (division by zero).

2. The choice of the number k of prototype that the network has to memorize (number of the Gaussian nodes) is the second weakness of the k-means algorithm. This problem of the network complexity is presented by the Fig. 2. We can see on this figure the influence of the parameter k in the prediction performances of the RRBF network. The complexity graph of the ANN is divided into three zones: a) an under training situation (zone1), b)a good generalization situation (zone 2), and c) an overtraining situation (zone 3). The second zone (zone 2 of good training) represents the good choice of the number k. in the literature, there is no formal technique describing the way to have this good training situation [1].

3. The random initial choice of the k center from all the input data is the third weakness. This might cause the instability of the results. Consequently we have to run the algorithm several times to obtain a good result. As showed in Fig. 3, we have different results at each execution cycle.

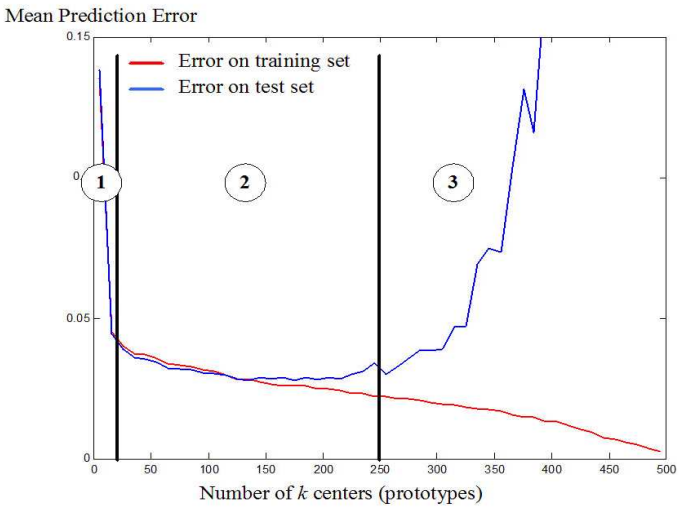


Fig. 2. Influence of the k parameter
in the prediction performances of the RRBF.

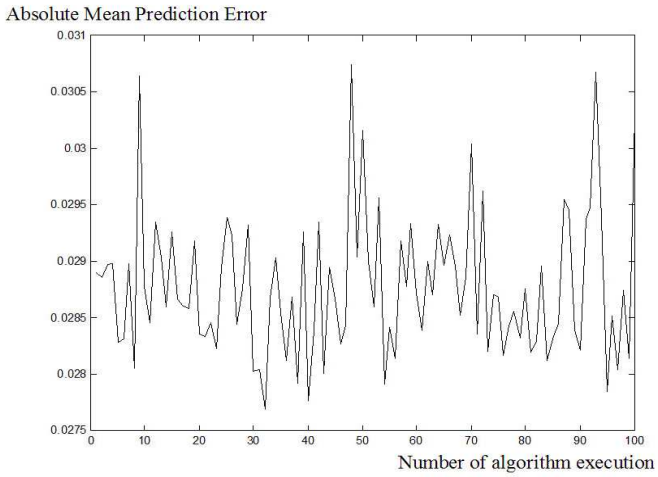


Fig. 3. Instability of the k-means technique.

In this paper we present a method meant to compensate the three weaknesses previously enumerated. We use the *Fuzzy Min-Max* technique [26], [27] to find the numbers of the k-center, and to initialize them. The organization of the remaining part of this paper is as it follows. Section 2 gives a review of the *Fuzzy Min-Max* technique and indicates how we use it to improve the performances of the k-means algorithm. The experimental studies and the results obtained are explained and discussed in section 3. Conclusion and suggestions for further work are presented in section 4.

2. *The Fuzzy Min-Max* algorithm

The instability problem of the k-means technique can be solved by the *Fuzzy Min-Max* (FMM) technique [26], [27]. This technique allows determining and initializing the k-centers in an iterative way. Thus the k-means technique can be boosted to converge into a good and stable result. The FMM technique is based on the use of hyper-cubes with fuzzy sets. A hyper-cube defines a region of the n-dimensional pattern space that has patterns with full class membership. The hyper-cube is completely defined by its minimum and maximum points, and its corresponding membership functions are used to create fuzzy subsets in the n-dimensional pattern space. FMM is an incremental learning system. It learns incrementally in a single pass through the data. It refines the existing pattern classes as new information is received. It also has the ability to add new pattern classes online. During this initialization phase, $n-$dimensional hyper-cubes are created. The hyper-cubes are limited by the maximal and minimal coordinates of each hyper-cube points. A *membership function* is defined with respect to the minimum and maximum points of a hyper-cube. It describes the degree to which a pattern fits into the hyper-cube. The membership function of a point for each hyper-cube is defined as following:

$$H_j(x, v_j, u_j) = \frac{1}{n} \sum_{i=1}^{n} [1 - f(x_i - u_{ji}) - f(v_{ji} - x_i)], \tag{4}$$

where the function f is defined as:

$$f(x) = \begin{cases} 1, & x > \eta \\ x/\eta, & 0 \leq x \leq \eta \\ 0, & x < 0 \end{cases} \tag{5}$$

and H_j is the membership degree of the point $\mathbf{x}$ to the hyper-cube j. This membership degree belongs to $[0, 1]$; x_i is the i^{th} dimension of the input vector $\mathbf{x}$; u_{ji}, v_{ji} is the maximal and minimal value of the i^{th} dimension for the j^{th} hyper-cube.

The η parameter is called the *sensitivity* of each hyper-cube. The numerical value is the same for all the hyper-cubes. This parameter determines the decreasing of the membership degree H_j (equation (4)) of a point according to its distance from the j^{th} hyper-cube. For small values of η, we obtain a steeply membership degree (we lose the fuzzy aspect). In this case, there is no overlapping between hyper-cubes. Conversely, for great value of η, we obtain a very fuzzy membership degree H_j. The overlapping between hyper-cubes is very high in this case. The authors of the algorithm do not give a formal way to determine the sensitivity parameter η. The only criterion is to minimize the overlapping between hyper-cubes (small values of η). To observe this criterion with a certain fuzzy aspect of the membership degree, we propose the following expression to calculate the value of the parameter η:

$$\eta = \min_i \frac{\max\limits_{\mathbf{x}_j \in \chi}(x_{ji}) - \min\limits_{\mathbf{x}_j \in \chi}(x_{ji})}{2 \times (N - 1)}, \tag{6}$$

where x_{ji} is the i^{th} dimension of the input vector $\mathbf{x}_j$ of the cluster χ; N is the number of the input points of all the data set χ.

Figure 4 shows an example of the calculation of the sensitivity parameter η for a two-dimensional problem with an input set χ containing two points $\{\mathbf{x}_1, \mathbf{x}_2\}$:

$$\eta_1 = \frac{x_{11} - x_{21}}{2 \times (2-1)} = \frac{x_{11} - x_{21}}{2} \quad \text{and} \quad \eta_2 = \frac{x_{12} - x_{22}}{2 \times (2-1)} = \frac{x_{12} - x_{22}}{2}.$$

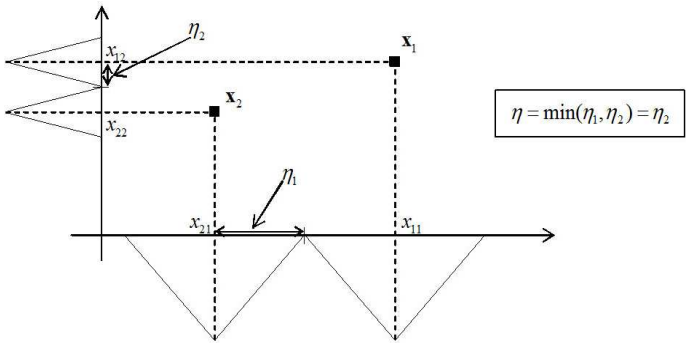


Fig. 4. Calculation of the sensitivity parameter η.

The Fuzzy Min-Max algorithm has three phases: a) the expansion of hyper-cubes, b) the overlap test and c) the contraction of the hyper-cubes. For the initialization phase of the k centers, we have used only the expansion phase to form the initial clusters. The steps used for the initialization of the k centers of the k-means algorithm are:

1. The initialization of the maximum and minimum points of the first hyper-cube with the first presented point.
2. The calculation of the membership degree of each input point by the Eq. (4).
3. The expansion of the hyper-cube having the highest membership degree is done according to Eq. (4):

$$\sum_{i=1}^{n} (\max(u_{ji}, x_i) - \min(v_{ji}, x_i)) \leq n\theta, \tag{7}$$

where θ represents a parameter of the algorithm which controls the creation of new hyper-cubes. A small θ means more hyper-cubes will be created, and conversely, big θ means less hyper-cubes will be created. A great number of hyper-cubes means that the hyper-cubes can contain only a smaller number of patterns, which will increase the network complexity. A small number of hyper-cubes means that they can contain a larger number of patterns, and will decrease the network complexity. If a hyper-cube is expanded, the old minimum and maximum points of the hyper-cube will be replaced by the new minimal and maximal values.

4. If no hyper-cube is expanded, a new hyper-cube containing the new input data will be generated (condition of the equation (7) is not respected).

Once the entire training data are presented, k hyper-cubes are then created according to the parameter θ. The k centers are so initialized by this iterative method.

Figure 5 shows comparative results between the random initialization (classic k-means initialization) and the Fuzzy Min-Max initialization. This figure shows the input data set and the k-center obtained with the two methods. By the Fuzzy Min-Max technique, we have a certain uniformity of the k centers according to the cluster density. This Fuzzy Min-Max initialization is more stable than the classic initialization of the k-means technique. We have always the same result at each algorithm execution. But with the classic k-means technique, the k-centers are randomly initialized, so the final result is different at each algorithm execution.

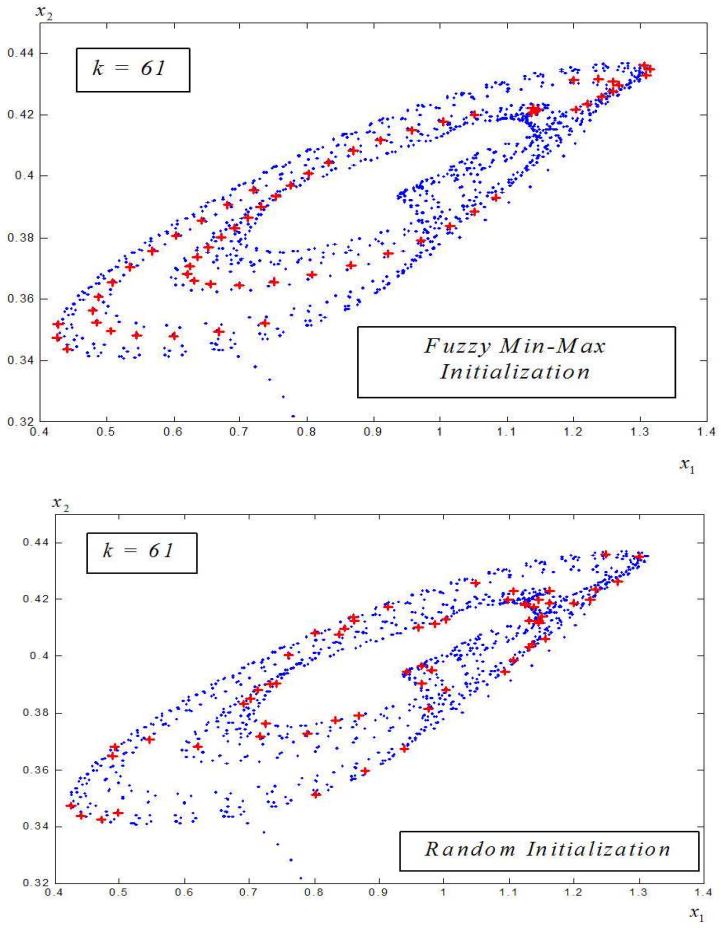


Fig. 5. Comparative results between a random initialization (classic k-means initialization) and the Fuzzy Min-Max initialization.

We propose the boosted k-means algorithm version:

1. Initialization of the Fuzzy Min-Max parameter θ,
2. Creation of the k hyper-cubes with one iteration of the Fuzzy Min-Max algorithm,
3. Initialization of the k-center μ_i *with the mean of each hyper-cube,*
4. Do until no change:
  4.1. Do until no change:
    4.1.1. Classify the training data $x \in \chi$ *to the nearest center* μ_i,
    4.1.2. Calculate the new means of each cluster $\{\mu_1, \mu_2, ..., \mu_k\}$.
  4.2. Eliminate the empty cluster

In Fig. 6 we study the influence of the variation of the parameter θ on the number k of the created centers. This figure shows these results obtained on the temporal Macky-Glass series prediction [6]. We can see that whatever θ variation, the number of created prototypes is situated in a zone bounded by both superior and lower borders (Fig. 6b). This zone is well situated in the good generalization zone. The k-means algorithm is then boosted to converge into this zone of good generalization. Thus it avoids the under training and overtraining zone. This version is less sensitive to the parameter setting delicate phase.

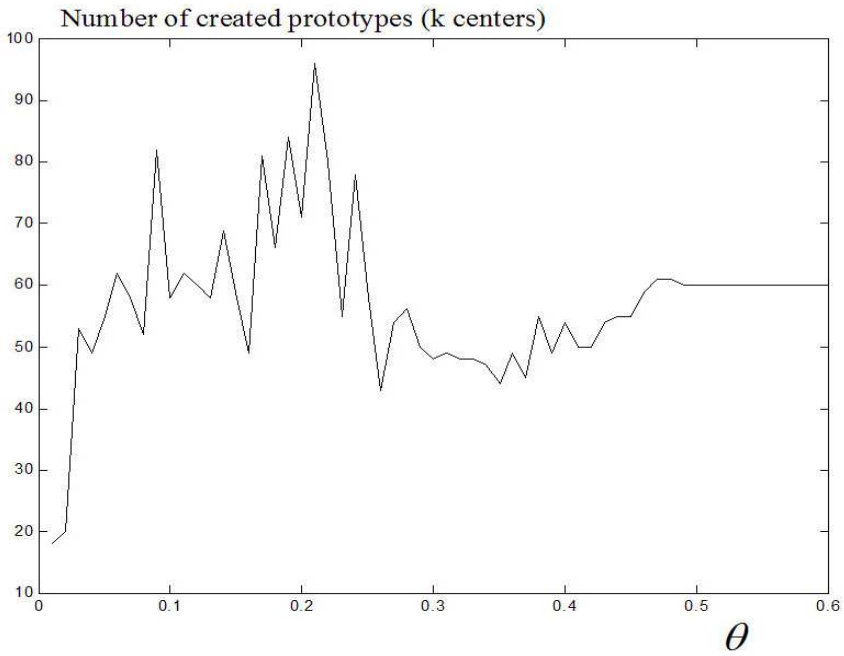


Fig. 6a

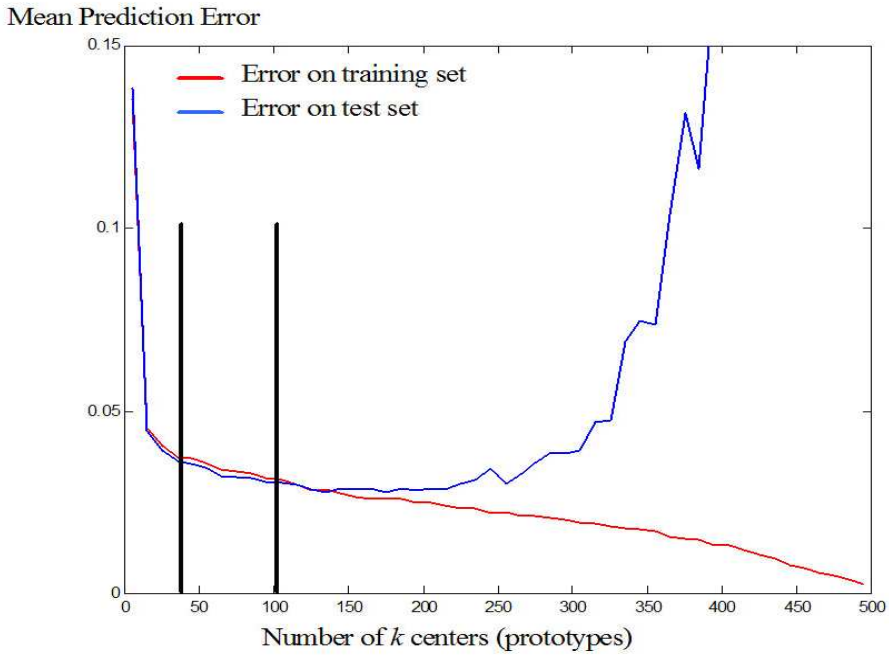


Fig. 6b

Fig. 6. The influence of parameter θ variation: a) number of created prototypes according to the variation of the parameter; b) the convergence zone limits of the boosted k-means algorithm.

3. Experimental Test of the Algorithm

In this section we present the results obtained on a prediction benchmark application: the *Box & Jenkins gas furnace* benchmark [29]. We have to predict the output concentration of CO_2 $y(t+1)$ of gas furnace from the concentration $y(t)$ and the input gas $u(t)$(see Fig. 7).

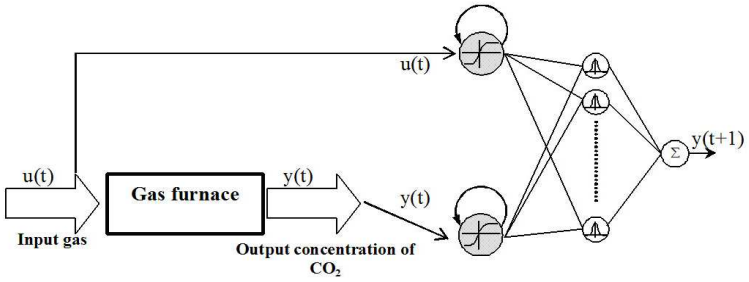


Fig. 7. Prediction of the output concentration of a gas furnace.

The training data base contains 300 values of $y(t)$ and $u(t)$. We have used the 100 first values for the training phase and the last 200 values for the test phase (Fig. 8).

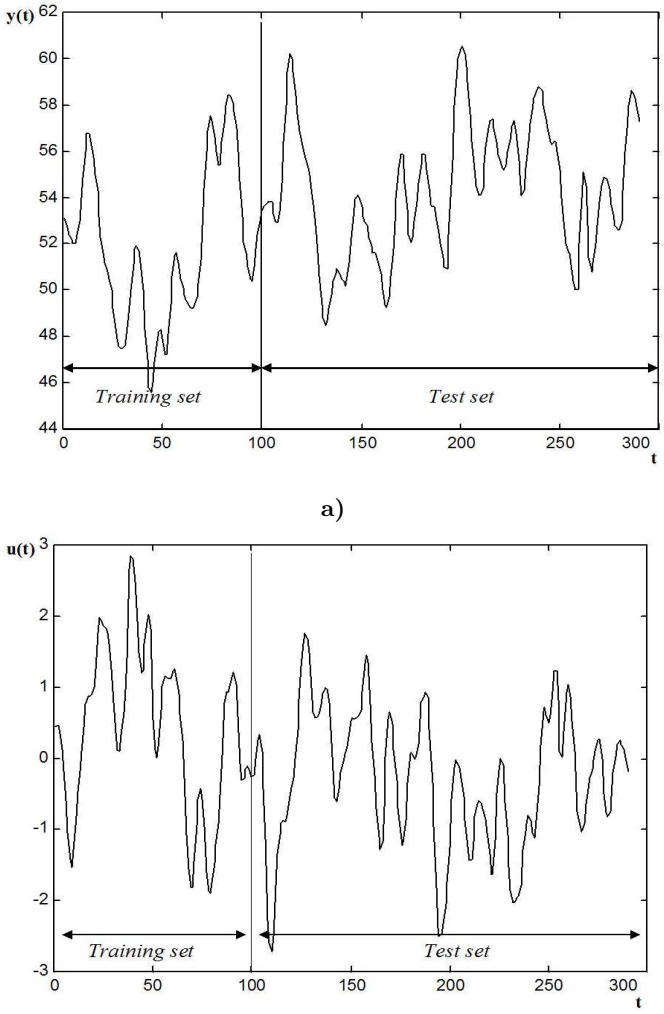
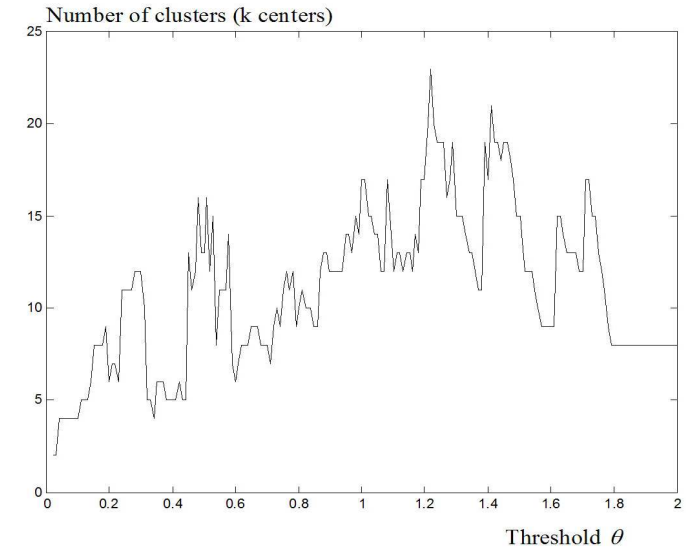


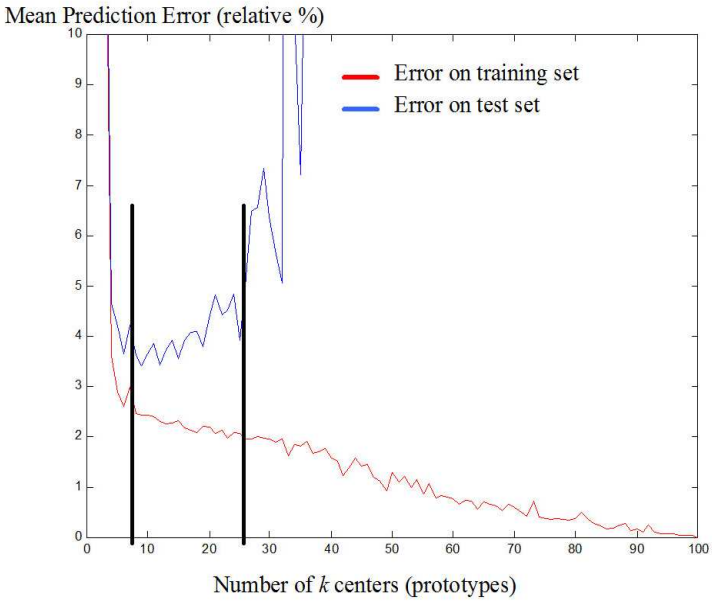
Fig. 8. The training data base: a) output gas furnace CO_2 concentration; b) input gas.

Figure 9 presents the relation between the parameter θ and the complexity of the neural network (number of the k-centers). As we see in Fig. 9a, if the value of the parameter θ is around 1, the number of the created prototypes is situated in a zone of a good generalization (bounded by both superior and lower borders). These values of the parameter θ force the RRBF network to converge into the good generalization zone. We then avoid both under training and overtraining zones. The

FMM initialization guarantees to k-means algorithm a good choice of the number k of the prototype. These k centers are also goodly initialized by the hyper-cubes of the FMM technique.



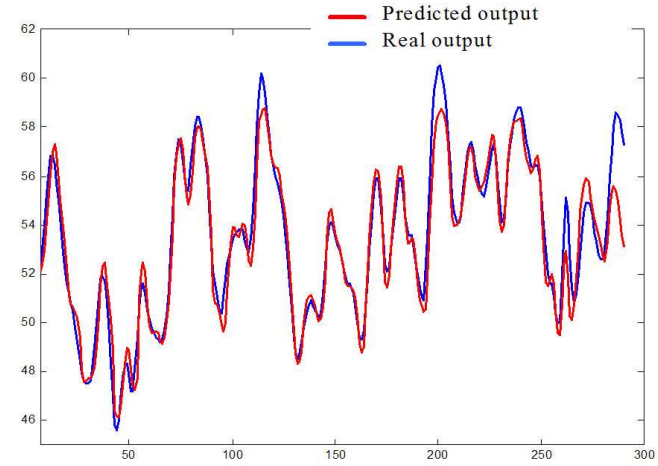
a)



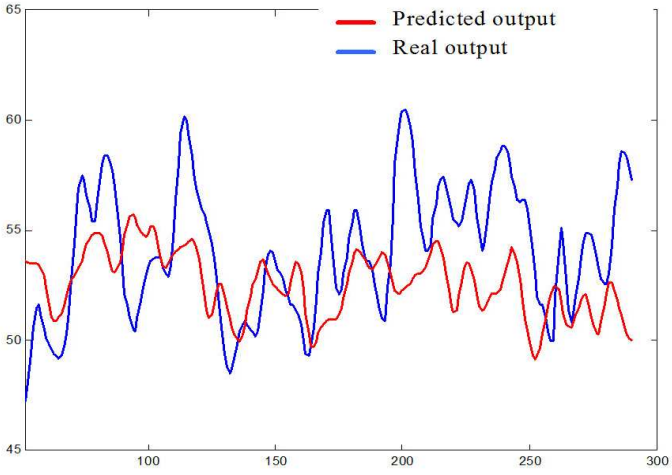
b)

Fig. 9. The influence of parameter θ on a) the number of created prototypes and b) the corresponding prediction error in both training and test data set.

Figure 10 presents the prediction results obtained by the RRBF network. Figure 10a shows a 5 steps prediction (prediction at t+5) and Fig. 10b shows a 50 steps prediction (prediction at t+50). Figure 11 presents the relative prediction error obtained by the classic version of the k-means for $k = 19$ (random initialization of the k-center) and the FMM technique. For the classic version of the k-means, we have run the algorithm 100 times, and we obtained different result each time. With the FMM initialization, we always obtained the same result. In Fig. 11 we can see that the mean prediction error with the FMM technique is close to one of the best results obtained by the random initialization.



a)



b)

Fig. 10. The prediction results obtained by the RRBF network with a local output feedback memory.

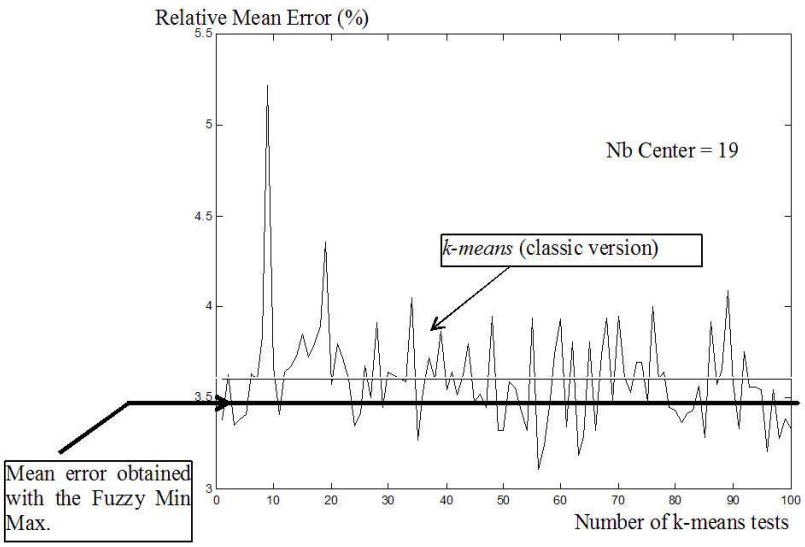


Fig. 11. Stability of the boosted k-means algorithm.

4. Experimental Validation of the Algorithm

For the two benchmarks described in this section, we have compared two ways of predicting the output system $\hat{x}(t+1)$. In the first one we do not use any prediction error. We have used the RRBF as presented in Fig. 12a. For the second one (Fig. 12b) we have calculated the prediction error $\varepsilon(t)$ between the output system $x(t)$ and the neural network predicted output $\hat{x}'(t)$. A proportional–integral–derivative controller (PID controller) attempts to correct this error and then outputs a corrective action that can adjust the final Neural Network prediction output $\hat{x}(t+1)$. In this section, this prediction structure is called RRBF_{Error}. The output of the RRBF_{Error} is then described by these equations:

$$\hat{x}(t+1) = \hat{x}'(t+1) + K_p\varepsilon(t) + K_i \int_0^t \varepsilon(\tau)\partial\tau + K_d \frac{\partial\varepsilon(t)}{\partial t}, \tag{8}$$

$$\varepsilon(t) = x(t) - \hat{x}'(t), \tag{9}$$

$$\hat{x}'(t+1) = \sum_{j=1}^{M} (w_{kj}\phi_j(\|\mathrm{x} - \mathrm{u}_j\|)). \tag{10}$$

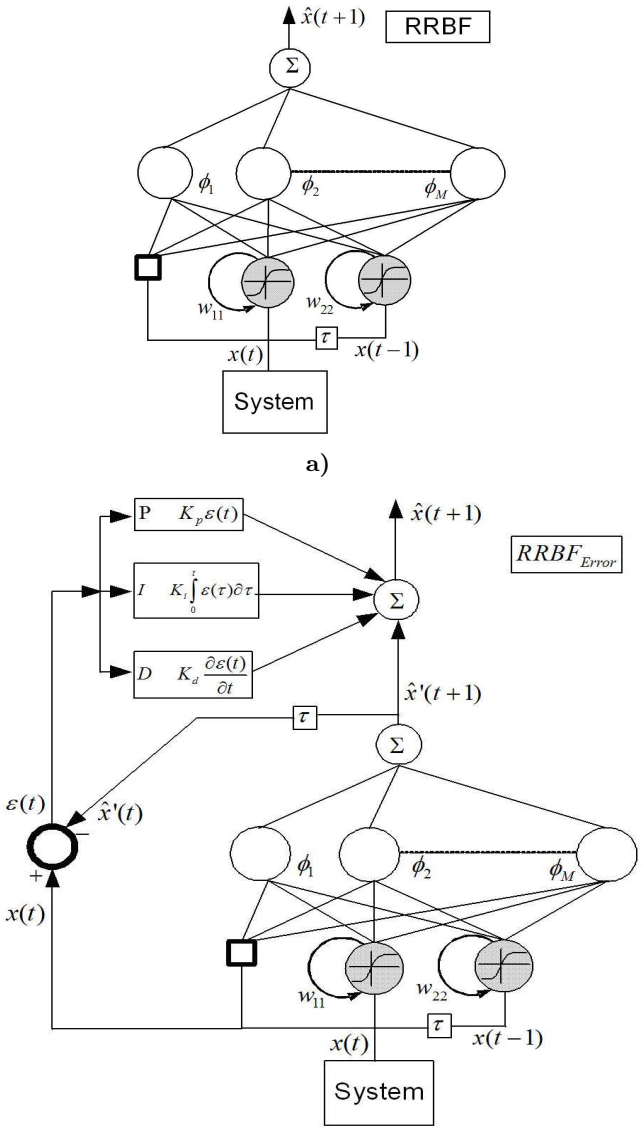


Fig. 12. The RRBF and $\text{RRBF}_{\text{Error}}$ structure for time series prediction.

The goal of the first time series (the Mackey–Glass differential equation [28]) is to predict the value of time series at some point in the future $x(t+n)$ by using past values. The second benchmark is the Box–Jenkins furnace data describes in the previous section. For the two benchmarks, we have tested 10 prediction horizons (n varying from 1 to 10), and the training data set for the two benchmarks contains 50 points. The two neural structures were trained by the k-means algorithm that was

boosted by the Fuzzy Min-Max as described in the previous sections. In all cases, the error measure used for evaluation is the mean square error (MSE) since it is the most commonly used measure found in literature. All data have been normalized by range [–1,+1]. Results in Table 1 and Table 2 show the best overall MSEs obtained respectively for both Box–Jenkins and the Makey-Glass with the two ANNs (RRBF and RRBF$_{Error}$) and for different horizons of prediction (t+n). Figures 13 and 14 show the MSE for these different horizons of prediction for the two benchmarks. We can see that for all the tests, the integrator parameter K_I which gives the best result is equal to zero (K_I=0). This is due to the summation done by the integrator that gives an increasing prediction error at each step. Figures 15 and 16 present the MSE for different number of the k centers of the Gaussian nodes. We can see that the RRBF$_{Error}$ is more stable than the RRBFn. The MSE of the RRBF model can highly vary with the variation of the number of the k center.

The training algorithm presented in this paper always gives a good initialization of the k-means algorithm. So, we don't have to initialize randomly the k-means algorithm. Figures 17 and 18 show some of the prediction results obtained with the best model for each ANN and for each prediction horizon (as presented in Table 1 and in Table 2). From all these results, it is obvious that the prediction performances of the RRBF are highly improved with the *proportional-derivative* controller (PD controller). For the two benchmark tests, the prediction results obtained with the RRBF$_{\text{Error}}$ are better than those obtained with the RRBF network.

Table 1. Box and Jenkins results

Horizon of prediction (t+n)	Neural Network	No. nodes	K_p	K_i	K_d	MSE test set
t+1	RRBF	16	-	-	-	9.2346585e-003
	RRBF$_{\text{Error}}$	10	0.8	0	0.5	2.4931081e-003
t+2	RRBF	12	-	-	-	2.0891934e-002
	RRBF$_{\text{Error}}$	9	0.5	0	1	1.3293116e-002
t+3	RRBF	15	-	-	-	2.1827862e-002
	RRBFError	31	0.7	0	0.5	1.8546084e-002
t+4	RRBF	27	-	-	-	1.9543758e-002
	RRBF$_{\text{Error}}$	16	0.2	0	0.2	2.1365814e-002
t+5	RRBF	5	-	-	-	2.5234884e-002
	RRBF$_{\text{Error}}$	6	0.8	0	0.9	2.1704984e-002
t+6	RRBF	4	-	-	-	4.2036604e-002
	RRBF$_{\text{Error}}$	7	0.4	0	0.1	3.7381127e-002
t+7	RRBF	7	-	-	-	7.7557329e-002
	RRBF$_{\text{Error}}$	6	0.5	0	0	7.5398560e-002
t+8	RRBF	7	-	-	-	1.2138848e-001
	RRBF$_{\text{Error}}$	4	0.3	0	0	1.1880591e-001
t+9	RRBF	7	-	-	-	1.7708602e-001
	RRBF$_{\text{Error}}$	8	0.5	0	0	1.5360799e-001
t+10	RRBF	7	-	-	-	2.1988363e-001
	RRBF$_{\text{Error}}$	5	0.6	0	0	1.7519635e-001

Table 2. Makey-Glass results

Horizon of prediction (t+n)	Neural Network	No. nodes	K_p	K_i	K_d	MSE test set
t+1	RRBF	4	-	-	-	1.1234558e-002
	$RRBF_{Error}$	7	1	0	0.9	4.2470656e-005
t+2	RRBF	4	-	-	-	3.1526859e-002
	$RRBF_{Error}$	7	1	0	0.9	1.6468035e-003
t+3	RRBF	2	-	-	-	4.6262829e-002
	RRBFError	6	1	0	0.9	8.1336288e-003
t+4	RRBF	2	-	-	-	6.4874782e-002
	$RRBF_{Error}$	2	0.7	0	0.9	2.1463932e-002
t+5	RRBF	2	-	-	-	8.5773478e-002
	$RRBF_{Error}$	2	0.6	0	0.9	3.9283539e-002
t+6	RRBF	2	-	-	-	1.0730872e-001
	$RRBF_{Error}$	2	0.5	0	1	5.8869717e-002
t+7	RRBF	2	-	-	-	1.2790076e-001
	$RRBF_{Error}$	2	0.5	0	1	8.2871004e-002
t+8	RRBF	2	-	-	-	1.4617843e-001
	$RRBF_{Error}$	2	0.3	0	1	1.0340636e-001
t+9	RRBF	2	-	-	-	1.6111025e-001
	$RRBF_{Error}$	2	0.3	0	1	1.2357237e-001
t+10	RRBF	2	-	-	-	1.7209771e-001
	$RRBF_{Error}$	2	0.1	0	1	1.3818208e-001

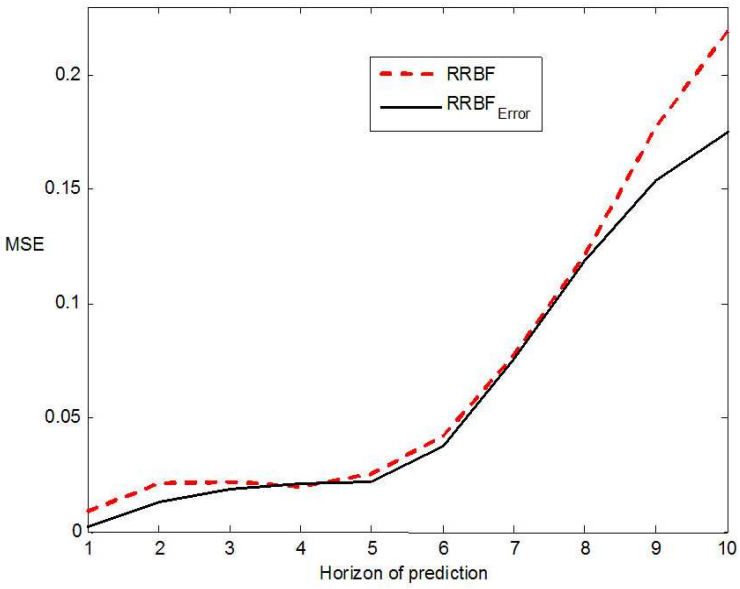


Fig. 13. Box and Jenkins results.

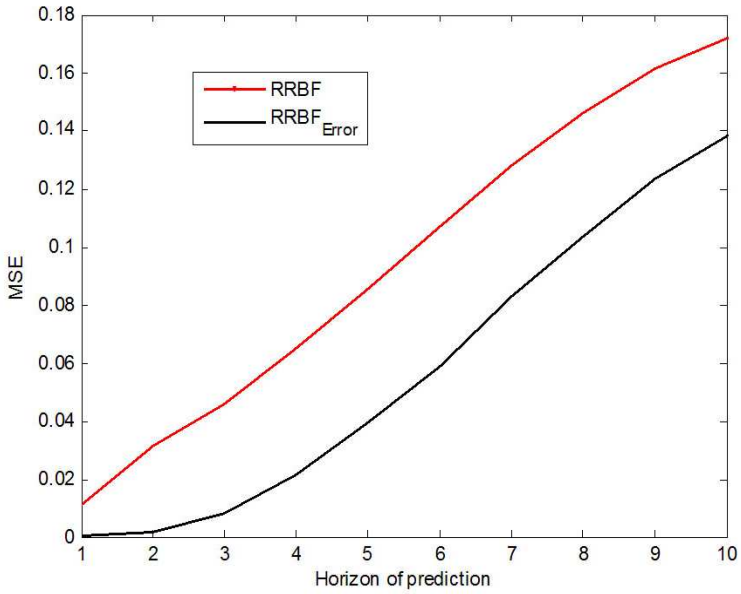


Fig. 14. Makey-Glass results.

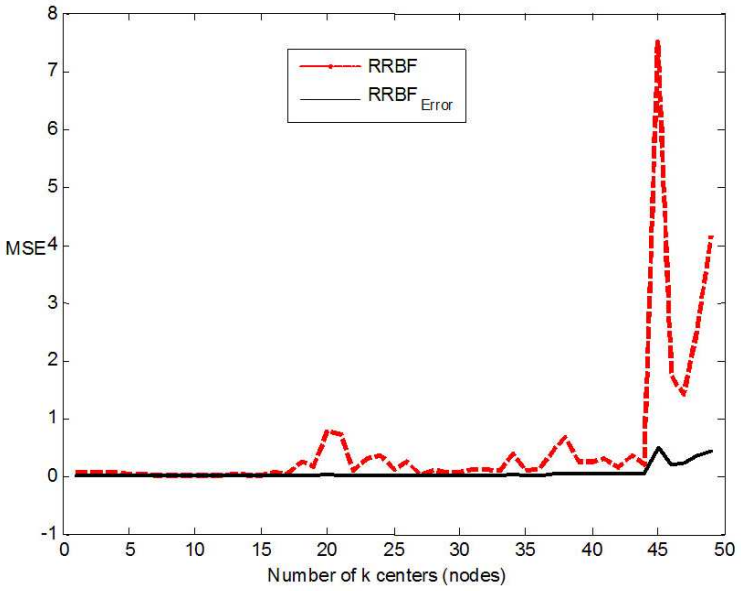


Fig. 15. Box and Jenkins results for (t+1).

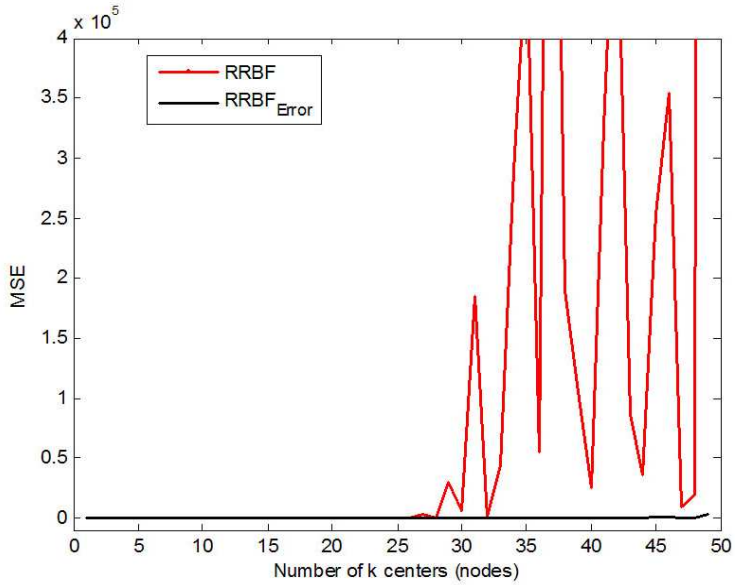
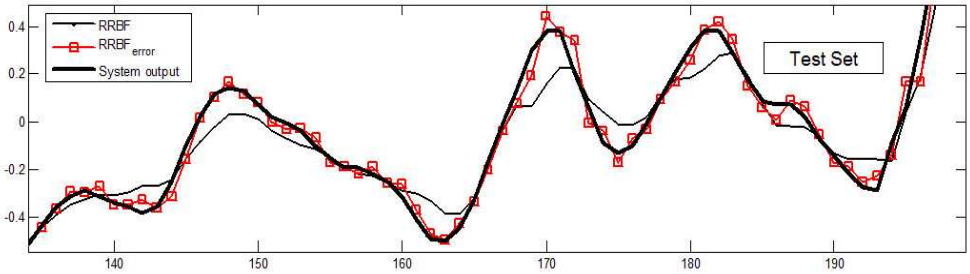
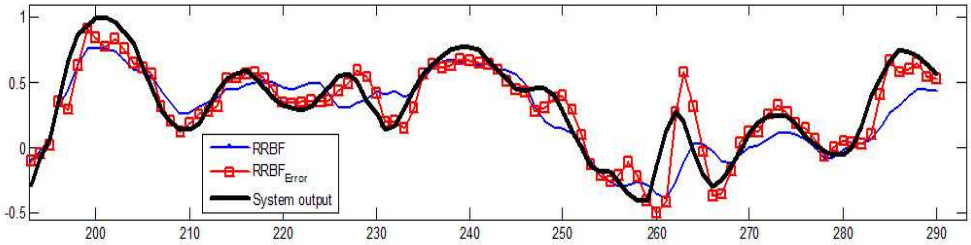


Fig. 16. Makey-Glass results results for (t+1).

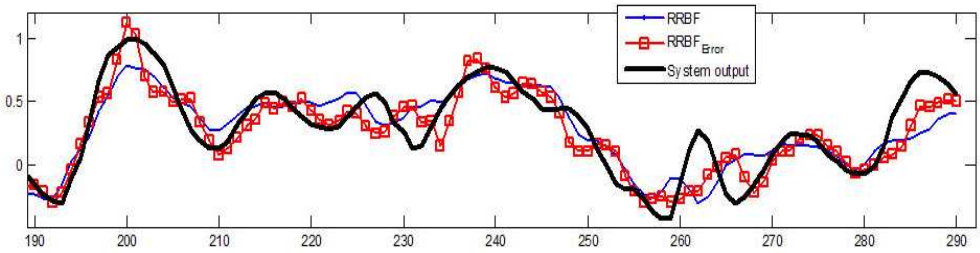
(t+1):



(t+2):



(t+3):



(t+4):

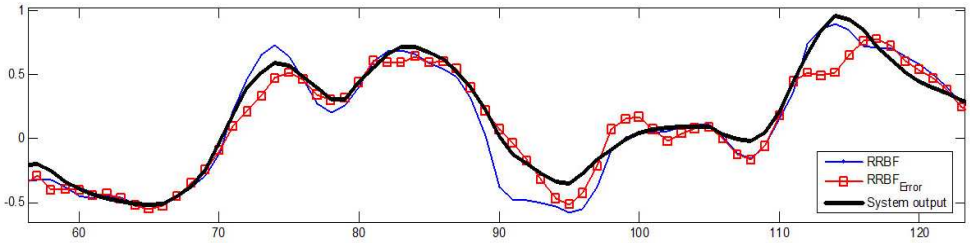
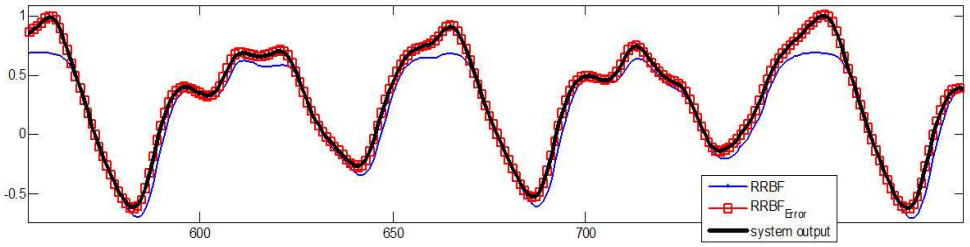
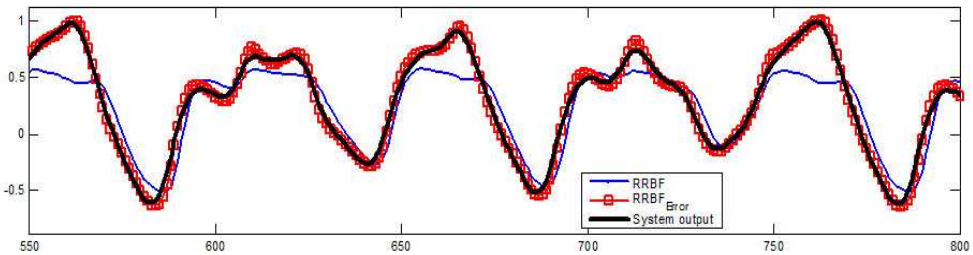


Fig. 17. Box and Jenkens results for different horizon prediction.

(t+1):



(t+2):



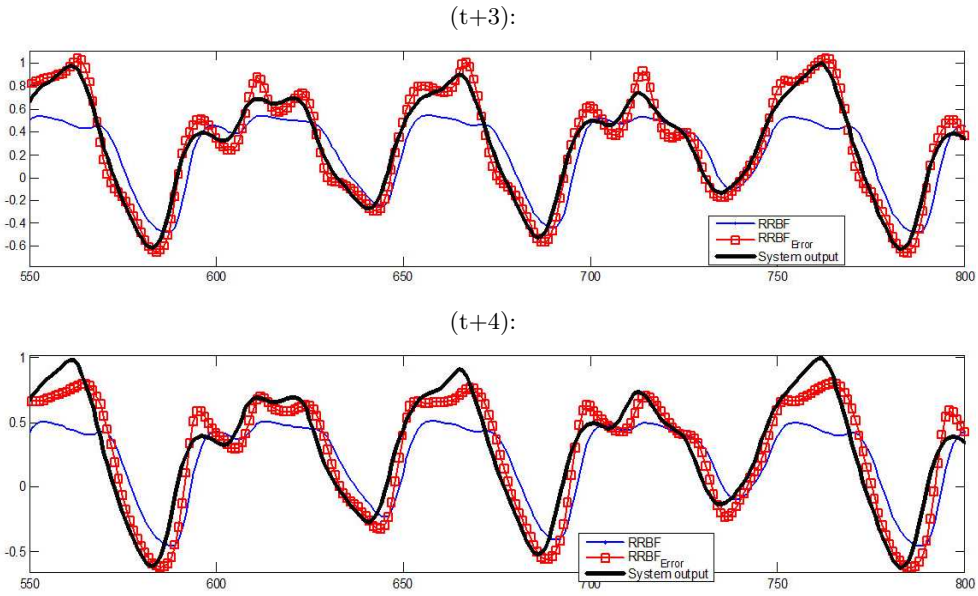


Fig. 18. Makey-Glass results for different horizon predictions.

5. Conclusion

In this paper we have presented an improved version of the k-means algorithm to have good prediction performances of the Recurrent RBF network. We have used the Fuzzy Min-Max technique to initialize the k centers. The k-centers are not obtained by a random way but by an iterative technique. We notice that the FMM initialization guarantees to the k-means algorithm the convergence to one of the best prediction results. We have then resolved the random initialization of the k-means algorithm.

Although the results obtained from the experiments are encouraging, more investigations with data set from different problem domains are needed to further ascertain the effectiveness of the proposed training algorithm. We have to compare the performances of the proposed algorithm with other techniques like the modified FMM network presented in [3].Also new applications in various fields such fault detection and diagnosis and problem recognition and model building in decision- making [5] are envisaged.

Note. A preliminary version of this paper was presented at the 2nd Mediterranean Conference on Intelligent Systems and Automation (CISA'09 [30]).

References

[1] CHANG F. J., LIANG J. M., CHEN Y. C., *Flood Forecasting Using Radial Basis Function Neural Network*, IEEE Transactions on Systems, Man and Cybernetics – Part

C: Applications and Reviews, **Vol. 31**, No. 4, November 2001.

[2] ZEMOURI R., RACOCEANU D., ZERHOUNI N., *Réseaux de neurones Récurrents à Fonction de base Radiales: RRFR/ Application au pronostic*, Revue d'Intelligence Artificielle, RSTI série RIA, **Vol. 16**, No. 03, 2002.

[3] QUTEISHAT A., LIM C. P., *A modified fuzzy min-max neural network with rule extraction and its application to fault detection and classification*, Applied Soft Computing, Science direct, Elsevier, 2007.

[4]] FILIP F. G., *Decision support and control for large-scale complex systems*, Annual Reviews in Control, **32**(1)(2008), pp. 61–70.

[5] FILIP F. G., *Sisteme suport pentru decizii (Decision Support Systems)*, 2nd ed., Editura Tehnica, Bucharest, 2007 (in Romanian).

[6] CHAPPELIER J. C., GRUMBACH A., *A Kohonen Map for Temporal Sequences*, Proceeding of Neural Networks and Their Application, NEURAP'96, IUSPIM, Marseille, mars 1996, pp. 104–110.

[7] CHAPPELIER J. C., *RST: une architecture connexionniste pour la prise en compte de relations spatiales et temporelles*, Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, January 1996.

[8] ELMAN J. L., *Finding Structure in Time*, Cognitive Science, **vol. 14**, June 1990, pp. 179–211.

[9] BARNSLEY M., *Fractals Everywhere*, Academic Press, Boston, 2nd edition, 1993.

[10] KOLEN J. F., *Fool's gold: Extracting finite state machines from recurrent network dynamics*, in Cowan J. D., Tesauro G. and Alspector J., editors, *Advances in Neural Information Processing Systems*, **vol. 6**, pp. 501–508, Morgan Kaufmann Publishers Inc., 1994.

[11] DEVANEY R. L., *An introduction to Chaotic Dynamical Systems*, Addison-Wesley, 1989.

[12] CASSANDRAS C. G., LAFORTUNE S., *Introduction to Discrete Event Systems*, 2nd Ed., Springer, 2007.

[13] TINO P., HORNE B. G., GILES C. L., COLLINGWOOD P. C., *Finite state machines and recurrent neural networks – automata and dynamical systems approaches*, in Dayhoff J., Omidvar O., editors, *Neural Networks and Pattern Recognition*, pp. 171–220, Academic Press, 1998.

[14] CASEY M., *The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction*, Neural Computation, **8**(6), pp. 1135–1178, 1996.

[15] GRAVES D., PEDRYCZ W., *Fuzzy prediction architecture using recurrent neural networks*, Neurocomputing, 2008.

[16] NEURMAIER A., SCHNEIDER T., *Estimation of parameters and eigenmodes of multivariate autoregressive models*, ACM Transactions on Mathematical Software **27**(1), pp 27–57, 2001.

[17] BARBOUNIS T. G., THEOCHARIS J. B., *A locally recurrent fuzzy neural network with application to wind speed prediction using spatial correlation*, Neurocomputing, 2006, doi:10.1016/j.neucom.2006.01.032.

[18] HARPHAM C., DAWSON C. W., *The effect of different basis functions on a radial basis function network for time series prediction: A comparative study*, Neurocomputing, **69**, 2006, pp. 2161–2170.

[19] ZHOU S. M., XU X.-D., *A new type of recurrent fuzzy neural network for modeling dynamic systems*, Knowledge-based Systems, **14**, 2001, pp. 243–251.

[20] ALIEV R. A., FAZLOLLAHI B., ALIEV R. R., GUIRIMOV B., *Linguistic time series forecasting using fuzzy recurrent neural networks*, Soft Computing, **12**(2), 2007, pp. 183–190.

[21] GAO Y., ER M. J., *NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches*, Fuzzy Sets and Systems, **150**, 2005, pp. 331–350.

[22] JACQUIN A. P., SHAMSELDIN A. Y., *Development of rainfall-runoff models using Takagi-Sugeno fuzzy inference systems*, Journal of Hydrology, **329**, 2006, pp. 154–173.

[23] VERNIEUWE H., VERHOEST N. E. C., DE BAETS B., HOEBEN R., DE TROCH F. P., *Cluster-based fuzzy models for groundwater flow in the unsaturated zone*, Advances in Water Resources, **30**, 2007, pp. 701–714.

[24] ZEMOURI R., RACOCEANU D., ZERHOUNI N., *Recurrent Radial Basis Function network for Time-Series Prediction, Engineering Applications of Artificial Intelligence*, The International Journal of Intelligent Real-Time Automation, journal IFAC - the, International Federation of Automatic Control, Ed. Elsevier Science, **vol. 16**, Issue 5–6, 2003, pp. 453–463.

[25] HARPHAM C., DAWSON C.W., *The effect of different basis functions on a radial basis function network for time series prediction: a comparative study*, Nerurocomputing, **69**, 2006, pp. 2161–2170.

[26] SIMPSON P.K., *Fuzzy min-max neural networks – Part I: Classification*, IEEE Transaction on Neural Networks, **Vol. 3**, No. 5, pp. 776–786, September 1992.

[27] SIMPSON P.K., *Fuzzy min-max neural networks – Part II: Clustering*, IEEE Transaction on Fuzzy Systems, **Vol. 1**, No. 1, pp. 32–45, February 1993.

[28] MACKEY M., GLASS L., *Oscillations and Chaos in Physiological Control System*, Science, pp. 197–287, 1977.

[29] BOX G.E.P., JENKINS G.M., *Time Series Analysis, Forecasting and Control*, Holden Day, San Francisco, 1970.

[30] ZEMOURI R., RACOCEANU D., ZERHOUNI N., MINCA E., FILIP F., *Training the Recurrent neural network by the Fuzzy Min-Max algorithm for fault prediction*, *AIP Conf. Proc., Intelligent Systems and Automation: 2nd Mediterranean Conference on Intelligent Systems and Automation (CISA'09)*, March 5, 2009, **Vol. 1107**, pp. 85–90.