Input Variable Selection using Parallel Processing of RBF Neural Networks (PP-RBFNNs)

M. Awad, H. Pomares, I. Rojas, O. Salameh, and M. Hamdon
Dept. of Computer Architecture and Technology, Granada, Granada, Spain
awad@atc.ugr.es
Faculty of information Technology University of Arab American University , Palestine
m.awad@aauj.edu

Abstract.

In this paper we propose a new technique focused on the selection of the important input variable for modelling complex systems of function approximation problems, in order to avoid the exponential increase in the complexity of the system that is usual when dealing with many input variables. The proposed parallel processing approach is composed of complete Radial Basis Function Neural Networks (RBFNNs) that are in charge of a reduced set of input variables depending in the general behaviour of the problem. For the optimization of the parameters of each RBFNN in the system, we propose a new method to select the more important input variables which is capable of deciding which of the chosen variables go alone or together to each RBFNN to build the parallel structure, thus reducing the dimension of the input variable space for each RBFNN. We also provide an algorithm which automatically finds the most suitable topology of the proposed parallel processing structure (PP-RBFNNs) and selects the more important input variables for it. Therefore, our goal is to find the most suitable of the proposed families of parallel processing architectures in order to approximate a system from which a set of input/output (I/O). So that the proposed (PP-RBFNN) outperforms other algorithms not only with respect to the final approximation error but also with respect to the number of computation parameters of the system.

Keywords:

Parallel Processing, input variable selection, radial basis function neural networks

1 Introduction

In many real world practical modelling problems, it is often possible to measure the value of many physical signals (variables), but it is not necessarily known which of them are relevant and required to solve the problem [1]. An excessively high computational complexity can occur when developing multivariate models for industrial or medical applications when the best set of inputs to use is not known. The main problems to face here are that when the input dimensionality increases, the computational complexity and memory requirements of the model increase (in some cases even exponentially); learning is more difficult with unnecessary inputs.

Neural networks can be defined as an architecture comprising massively parallel adaptive processing elements interconnected via structured networks. The main weakness of a neural network lies in its totally flat structure. A direct consequence of such structural simplicity is often a huge network, with an excessively large number of hidden units. One effective solution is to incorporate proper parallel processing structure into the network. Parallel processing structures have a very rich variety of applications in computing since they provide representations that can be composed, modified, and manipulated in a very flexible way [2,7,8,9].

The main problem to solve is that when the number of input variables increases, the number of parameters usually increases in a very rapid way, even exponentially. This phenomenon named the

curse of dimensionality [5] prevents the use of the majority of conventional modelling techniques and forces us to look for more specific solutions. To deal with this problem, input variable selection (IVS) procedures try to reduce the dimension of the input variable space, identifying and removing as much irrelevant and redundant data as possible, thus reducing the dimensionality of the data and allowing learning algorithms to operate faster and more effectively.

Input variable selection (IVS) has been researched intensively and has been applied to various problems such as data mining, knowledge discovery, pattern recognition, etc. One of the most popular methods used to select input variables is principal component analysis (PCA*).* Several authors have also worked to select the most important input variables in function approximation problems. Pomares et al in [3] presented a method to obtain the structure of a complete rule-based fuzzy system for specific approximation accuracy of the training data, deciding which input variables should be taken into account how many membership functions are needed in every selected input variable in order to reach the approximation target. The main drawback of that method is that it only could be applied to grid-based fuzzy systems with a limited number of input variables. Vehtari and Lampinen in [1] proposed to use posterior and marginal posterior probabilities obtained via variable dimension Markov chain Monte Carlo methods to find out potentially useful input combinations and to do the final model choice and assessment using the expected utilities computed by using the cross-validation predictive densities. Also noteworthy is the work made by Chen and Wang in [4], who proposed that for a given set of input and output variables, a fuzzy partition associating fuzzy sets with each input variable.

In our particular case, parallel processing architectures will be used to provide a suitable construction of parallel processing Radial Basis Function Neural Networks (PP-RBFNNs) which improve significantly the performance of complex function approximation problems. In this paper we show how our PP-RBFNN is capable of modelling complex systems without the above mentioned problems inherent to the increase of the number of input variables. For that purpose, we propose IVS method which tries to relate every dimension of the input data to the output target (as a function of one dimension) and divides the data of this dimension into parts. For each of these parts the distance is calculated between the maximum and minimum values of the output that belong to the input data of each dimension in each part and the average of all the distances in all parts. When the average has small value; the variable is more important and must be selected. The variables with big average are variables of noise and should be eliminated. The number of RBFNNs depends on the number of these variables and which of these go alone or together in a RBFNN. The process of deciding which of the variables go alone or together depends, in general, on the calculation of the distance variance of each variable or set of variables related to the output target.

We also propose an algorithm which automatically finds the most suitable topology of PP-RBFNN structure and selects the important input variables for it. Therefore, our goal is to find the most suitable of parallel processing architectures in order to approximate a system from which a set of input/output (I/O) data has been extracted.

The paper is organized as follows. *Section 2* describes the basic building modules and the parallel processing structures of RBFNN. *Section 3* presents the new procedure for IVS for our PP-RBFNNs. *Section 4* provides a method to select groups of input variables and the number of RBFNNs. *Section 5* presents the method of parameters optimization of each RBFNN. Finally, *Section 6* presents examples of how the proposed methodology is capable of finding the most suitable PP-RBFNN architectures with best final approximation error and less number of computation parameters of the system.

2 Architecture of the PP-RBFNN

In classical RBFNNs every neuron in the hidden layer receives all the input variables of the network. Nevertheless, the interconnections in the PP-RBFNN structure between input variables and the hidden layer are limited and located. The advantage of the PP-RBFNN structure consists of the fact that the problem is divided into many problems that are connected in parallel. Every problem is

presented a RBFNN. All the RBFNNs have a total output that is the output of the PP-RBFNN structure. This division of the system limits the quantity of the information of the previous layer. In general, to construct a PP-RBFNN structure to solve problems of function approximation two basic steps are needed:

- The identification of its structure. The number of RBFNNs depends on the number of the selected input variables and on which of these variables go alone or together to each RBFNN of the PP-RBFNN system).
- The estimation of the parameters of every RBFNN (centres $\vec{c}^{S}$, radius r^{s} and weight w^{S}, and the RBF in each RBFNN, and the calculation of the total output *F(x)* of the PP-RBFNN.

Fig. 3 presents the proposed parallel processing RBFNN system. Each one of the nodes of the figure is a RBFNN (see Fig. 2). RBFNNs can be seen as a particular class of Artificial Neural Networks (ANNs). The basic architecture of an RBFNN is a 3-layer network. The output of the net is given by the following expression:

$$F(\vec{x},\Phi,w)=\sum_{i=1}^{m}\phi_i(\vec{x})\cdot w_i \tag{1}$$

where $\Phi=\{\phi_i : i=1,...,m\}$ are the basis functions set and *wi* the associate weights for every RBF. The basis function ϕ can be calculated as a Gaussian function using the following expression:

$$\phi(\vec{x},\vec{c},r)=\exp\left(\frac{\|\vec{x}-\vec{c}\|}{r}\right) \tag{2}$$

where $\vec{c}$ is the central point of the function ϕ and r is its radius.

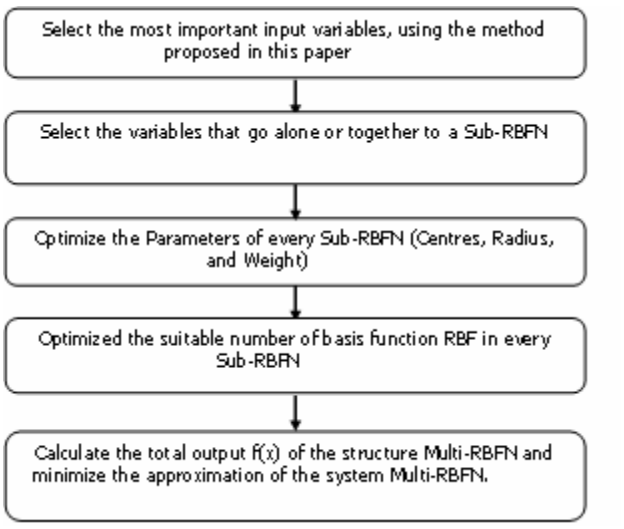


Fig. 1: Principal steps of the proposed algorithm

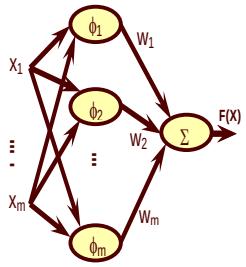


Fig.2: Radial Basis Function Neural Network

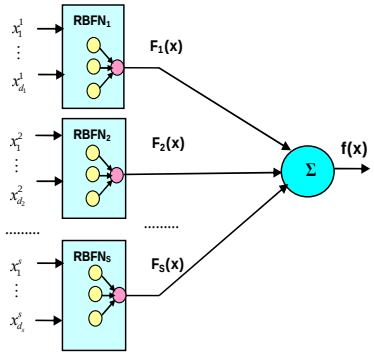


Fig.3: parallel processing structures PP-RBFN

Each subset of the input variables $\{x_1,\ldots,x_d\}$ (where d is the number of the dimensions of the input data space) can be used as the inputs of each RBFNN. Every group of the input variables is used as input of each RBFNN. These inputs are selected using our IVS procedure. Every RBFNN receives variables and implements the process of optimization of the parameters of every RBFNN (centres $\vec{c}^{\,s}$, radii r^s). When the parameters of centres $\vec{c}^{\,S}$ and radii r^s of each RBFNN have been optimized, a method of linear optimization is used to find the values of the weight *w*, which depends on the total output *f(x)* of the system PP-RBFNN, which minimizes the cost function calculated on the set of data I/O.

The optimization of the weight does not depend on every output of every RBFNN $\{F_1(x),\ldots,F_S(x)\}$, but it depends on the total output of the PP-RBFNN system, and must be calculated in the linear form as in the following expression:

$$f(\vec{x}, \Phi, w) = \sum_{s=1}^{S} \sum_{i=1}^{m_s} \phi_i^s(\vec{x}) \cdot w_i^s \qquad (3)$$

where ϕ_i^s are the i-th basis functions of the s-th RBFNN, and w_i^s is its weight.

Several PP-RBFNN structures can be obtained for any given problem from a set of input variables. For example, for a 4-input problem $\{x_1,\ldots,x_4\}$, many possible different architectures can be obtained, the simplest when each input variable forms a single set (See Fig.4.a), and the most complicated when all input variables are used in the only RBFNN (See Fig.4.d).

To gain an insight of how the PP-RBFNN configuration affects the number of actual parameters of the system, let us recall that the total number of parameters in every RBFNN is equal to $m \cdot (d+2)$, where *m* is the number of RBFs, and *d* is the number of input variables. Table 1 shows the number of parameters used in each one of the architectures of Fig.4 using (for fair comparison) a total number of 24 RBFs for each one. We can see how even for this simple example with only 4 input variables to share, the differences can be notable (the number of parameters can be doubled). The PP-RBFNN structure is thus capable of decreasing the number of parameters to optimize, provided that the selected structure is the most suitable one for the given set of I/O data.

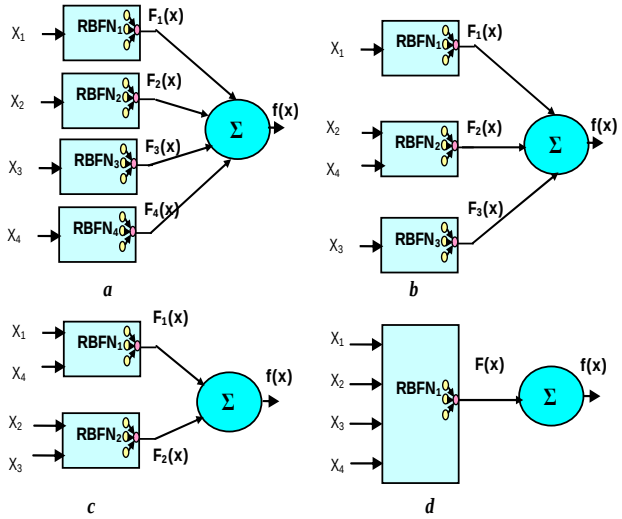


Fig.4. Different topologies of parallel processing PP-RBFNNs. a) 4 RBFNNs with one input variable for each one b) 3 RBFNNs with one and two input variables for each one. c) 2 RBFNNs with two input variables for each one. d) 1 RBFNNs with all the input variable set.

Fig #	RBFNN #	RBF # in each RBFNN	Var # in each RBFNN	Parm # in each Sub-RBFN	Parm # in PP-RBFN
2*a*	4	6	1	18	72
		6	1	18	
		6	1	18	
		6	1	18	
2*b*	3	6	1	18	84
		12	2	48	
		6	1	18	
2*c*	2	12	2	48	96
		12	2	48	
2*d*	1	24	4	144	144

Table 1. Number of parameters between different architectures PP-RBFNN.

In this paper we are concerned exclusively with the selection of the most suitable PP-RBFNN structure. However, some remarks could be made about the optimization of the rest of the parameters of the net, i.e. RBF centres, RBF radii and RBF weights. To optimize the centres of each RBF of each RBFNN, it is common to use clustering algorithms such as the one presented in [13]. For the radii, we used k-nearest neighbour technique [10]. Once the parameters of centres and radii of each RBFNN has been initialized we can use a linear optimization method for optimizing the values of the weights that minimize the least square errors.

3 Input Variable Selection for the PP-RBFNNs

An input variable selection method tries to reduce the dimension of the input variable space and creates a new input variable set, thus identifying and removing as much irrelevant and redundant data as possible, which reduces the dimensionality of the data and allows learning algorithms to operate faster and more effectively.

The curse of the dimensionality [11] refers to the exponential approximation of the hyper-volume as a function of dimensionality. RBFNN can be planned as interrelations of input space to output space, it has to cover or represent each part of its input space in order to know how that part of the input space should be mapped. Covering the input space take resources, and in the most general case, the amount of resources needed is proportional to the hyper-volume of the input space. The exact formulation of resources and part of the input space depends on the type of the network and should probably be based on the concepts of information theory and differential geometry [11]. Input variable selection fundamentally affects the severity of the problem, as well as the selection of the neural network model [12].

Our method considers a simple calculation to select the input variables. The selection of the input variables is done using the following steps:

1) Relate each possible input dimension of data $\{x_1, \ldots, x_d\}$ with the dependent variable y (as a function in one dimension) as:
$$\{(x_1, y), (x_2, y), (x_3, y), \ldots, (x_d, y)\} \quad (4)$$
2) Divide the data of each dimension into *P* parts as:
$$\left\{P_i^{j-1} \le \left(\vec{x}^k\right)_i < P_i^j\right\} \; k = 1, \ldots, n; i = 1, \ldots, d; j = 1, \ldots, p \quad (5)$$
where *n* is the number of data of I/O, $\left(\vec{x}^k\right)_i$ is the component ith of the input vector kth.
3) Associate the data of each part *P* to corresponding output data as:
$$\left\{\left(\vec{x}^k\right)_i, y^k\right\} \Big/ P_i^{j-1} \le \left(\vec{x}^k\right)_i < P_i^j \quad (6)$$
4) Use the Kalman filter to smooth the vectors of the maximum and minimums in each part, and calculate the distance D_i^j between the maximum and the minimum values of the output in each partition of the input variable x_i:
$$D_i^j = \max(y_j^k) - \min(y_j^k) \;\; j = 1, \ldots p \quad (7)$$
5) Finally, for each input variable x_i we calculate the mean of distances $\bar{D}_i$. The smallest $\bar{D}_i$ the most important input variable for the problem. Fig.5 presents, in a schematic way, the general

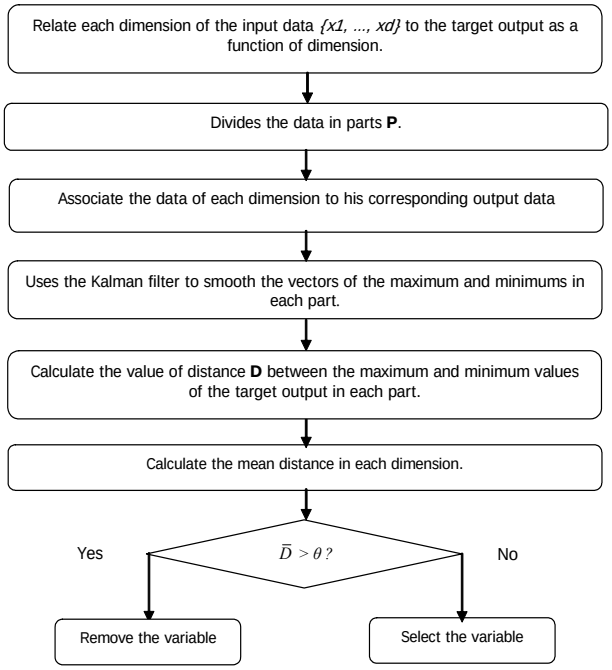


Fig.5: General description of the IVS method

description of the proposed IVS method. For all the parts the average of the distance is calculated $\bar{D}$.

4 Selection groups of input variables and the number of RBFNNs

This process depends on the function or the problem that we try to approximate. In general, every function is represented by forms of summation and/or multiplication and/or division and/or subtraction between its variables. The proposed PP-RBFNN structure tries to add linearly the output of every RBFNN to have the total output of the PP-RBFNN system. For this, the variables that come multiplied or divided and have not been eliminated by the calculation of the mean distance go together to one RBFNN. Any variable multiplied or divided by other/s variables does not produce a big change in the value of the variance of variable in the interval data, which we will always normalize in the interval [0,1]. The variables that come added or subtracted to other variables and have not been eliminated by the calculation of the mean distance go alone to one RBFNN. Any variable added or subtracted by other variables produce clear change in the value of the variance. The variance of the distance is calculated as:

$$\mathrm{var}(x_i) = \frac{\sum_{j=1}^{p} (D_j^i - \bar{D}_j)}{p-1} \tag{8}$$

The process of selecting which of the input variables must go alone or together to each RBFNN depends on the value of the variance of the distance between the maximum value and the minimum value in each partition. The variables that have a value of variance less than threshold variance will be selected to go to RBFNN. The task of analyzing the data begins with each of selected variables related to target output, and the variables with variance less than the proposed threshold value as variables that must go alone in a RBFNN. The variables that have not been selected in the first phase are analyzed in the next phases which take all possibilities of joining these variables, realizing every possible set of two, three, four, etc

5 Parameters optimization of each RBFNN

In the proposed system we use a new supervised method of clustering for initializing the values of the centres $\vec{c}^S$ in every RBFNN. This algorithm incorporates the information regarding the target output for every input vector of the set of training, and calculates the error provoked by each cluster in the output of the function or the problem that we want to approximate using a RBFNN. The number of clusters will increase in zones where the cluster provokes bigger error depending on the process of migration of the clusters that have minor error to zones of clusters that have bigger error and a process of local displacement that tries to allocate the data to the most nearby cluster [13].

When the centres values of every RBFNN are determined, the following step is to fix the values of the radius r^S of every basis function to cover all data. For that purpose we use a heuristic algorithm of k nearest neighbour*s (Knn)* [10].

Once the values of the centres $\vec{c}^S$ and radius r^S of the RBF have been optimized by means of the previous methods, every RBFNN will be a linear model and the set of weight w^S depends linearly on the samples of the set of training. In the PP-RBFNN system, the weight *w* is optimized depending on the total output. The calculation of the total output *f(x)* is the linear sum of all the output of each RBFNN *{F_1(x), …, F_S(x)}*.

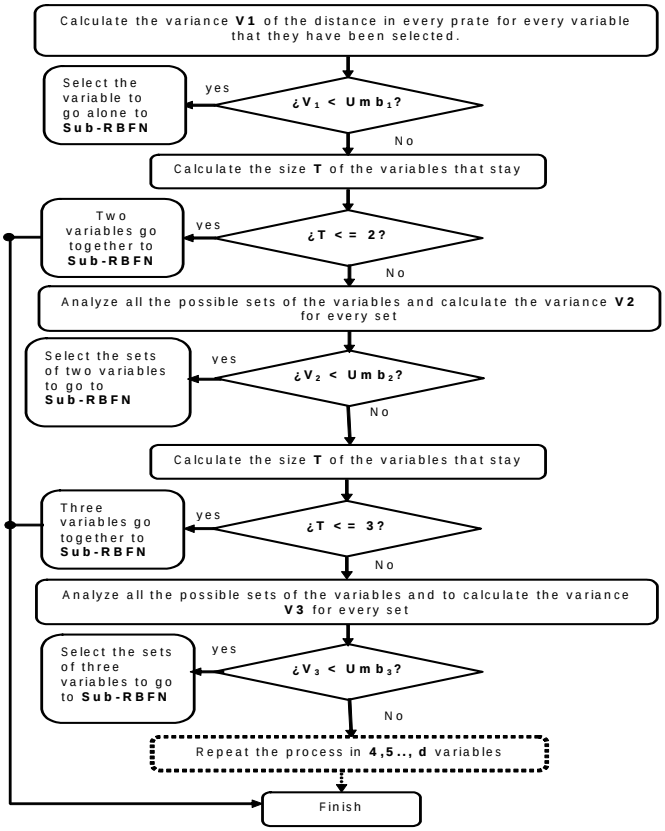


Fig.6: Process of selecting the variables that go alone or together to each RBFNN

The learning process is guided by the minimization of a function of error calculated as:

$$Er_n^d = \frac{1}{2}\sum_{j=1}^{d}\sum_{i=1}^{n}\left(f(\vec{x}_i^d, \Phi, w) - y_i\right)^2 \tag{9}$$

where $f(\vec{x}_i^d, \Phi, w)$ is the total output *f(x)* of the system, and y_i is the real output. The target of this phase is to find the optimal weight to calculate the total output and the error of approximation. To calculate the matrix of the weight w_m^s the following expression is used:

$$w_m^s = G \; Y \tag{10}$$

where *G* is the pseudo-inverse matrix of the activation matrix φ_m^s. This matrix can be calculated by means of methods of resolution of linear equations. In this algorithm we use the singular values decomposition (SVD) to solve this system of linear equations and assign the weight w^s for each RBFNN to calculate the output for each of them.

According to some methods the number of radial functions can be fixed priori or determined incrementally or decrementally. In the proposed system we use the incremental method to determine the number of RBF depending on the data test error that the system produces, which means, increase in each iteration only 1 RBF in one of RBFNN until the there is no improvement in test error during several iterations.

6 Simulation examples

In this section different examples are given to verify the procedure in the proposed algorithm. Two types of results are presented:

- The structure of the system PP-RBFNN

- The results of the validity of the algorithm in approximate functions from samples of I/O data, compared with results of a typical RBFNN that receives all the variables of the function and with other methods proposed in the bibliography.

The results are obtained in 5 executions; **{RBF}** the set of radial functions used in each RBFNN. **#Param** is the number of parameters. $\mathbf{NRMSE_{Tr}}$ is the normalized mean squared error of the training and $\mathbf{NRMSE_{Test}}$ is the normalized mean squared error of the test.

A. First Example $f_1(x)$

Suppose we take an example with 6 possible input variables to choose from. Let us consider a set of 20000 I/O data pairs randomly taken from the function.

$$f_1(x)=10\sin(\pi x_1 \cdot x_2)+20(x_3-0.5)^2+10x_4+5x_5+0x_6 \quad x_1,x_2,x_3,x_4,x_5,x_6 \in [0,1] \tag{11}$$

where each input variable is defined in the interval [0,1]. The proposed algorithm selects the ideal architecture of the system for the function $f_1(x)$, depending on the value of the variance threshold after analyzing every variable Fig.7.

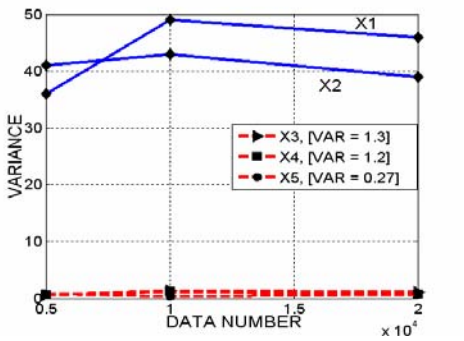
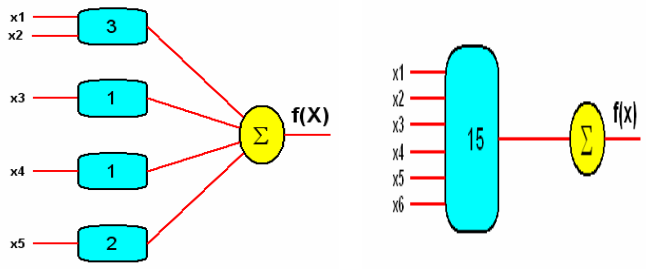


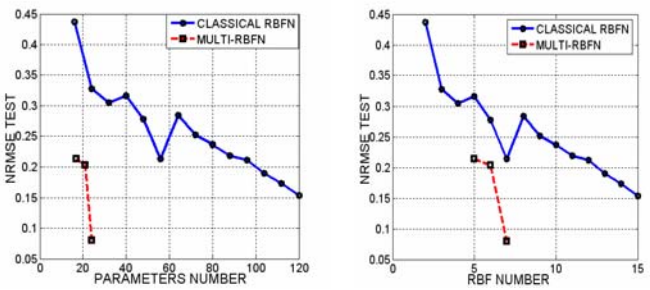
Fig.7. The variance for each variable in $f_1(x)$

In the function $f_1(x)$ few variables must go alone to RBFNN and the subset of the rest goes to other RBFNN, as in Fig.8a.



(a) **(b)**

Fig.8. (a) Structure PP-RBFNN selected by the algorithm. (b) Structure of a classic RBFNN for the current function



(a) **(b)**

Fig. 9. Comparison of the result of PP-RBFNN system and Classical RBFNN. (a) In the number of parameters. (b) In the number of RBF.

PP-RBFNN algorithm					
{ RBF }	**# Param**	**$NRMSE_{Tr}$**	**Std**	**$NRMSE_{Test}$**	**Std**
{2 1 1 1}	17	0.212	2E-3	0.214	1E-4
{1 2 1 1}	16	0.246	6E-3	0.252	4E-4
{1 1 2 1}	16	0.238	1E-2	0.243	5E-3
{1 1 1 2}	16	0.241	1E-2	0.246	6E-4
{3 1 1 1}	24	0.198	2E-1	0.204	1E-4
{2 2 1 1}	18	0.221	9E-3	0.225	1E-2
{2 1 2 1}	18	0.209	2E-3	0.216	6E-3
{2 1 1 2}	18	0.212	1E-3	0.216	1E-4
{4 1 1 1}	33	0.183	1E-2	0.189	8E-3
{3 2 1 1}	25	0.146	8E-2	0.147	3E-2
{3 1 2 1}	25	0.075	5E-3	0.084	3E-3
{3 1 1 2}	25	0.080	3E-3	0.088	2E-3
Classical RBFNN					
RBF	**# Param**	**$NRMSE_{Tr}$**		**$NRMSE_{Test}$**	
2	16	0.428		0.437	

3	24	0.331	0.328
4	32	0.301	0.305
5	40	0.316	0.316
6	48	0.279	0.278
7	56	0.213	0.214
8	64	0.284	0.284
9	72	0.249	0.252
10	80	0.231	0.237
11	88	0.211	0.219
12	96	0.206	0.212
13	104	0.179	0.190
14	112	0.153	0.173
15	120	0.144	0.154

Table.2 NRMSE of training and test obtained by the proposed algorithm and by classic RBFNN for the function $f_1(x)$

A. Second Example $f_2(x)$

In this example, the results obtained by the algorithm and other methodologies proposed in the bibliography are compared using the function $f_2(x)$.

$$f_2(x)=1.3356(1.5(1-x_1)\,.e^{(2x_1-1)}\cdot sen(3\pi(x_1-0.6)^2)+..$$
$$e^{(3(x_2-0.5))}\cdot sen(4\pi(x_2-0.9)^2)),\ x_1,x_2\in[0,1] \qquad (12)$$

They are compared with methods usually used to solve the problem of functional approximation, as methods presented in [14, 15, 16, 17]. Table.4 presents the results obtained by these methods for the function $f_2(x)$ and compared with other methods in [18, 19, 20]. In the function $f_2(x)$ each one of the variables go alone to each RBFNN, as in Fig .10.

As seen from Table IV, the result of the PP-RBFNN outperforms other algorithms.

Algoritmo	***m***	**Test NRMSE**	**# Param**
MLP [16]	15	0.096	**60**
PP [14]	-	0.128	**-**
CTM [16]	-	0.170	**-**
MARS [15]	-	0.063	**-**
ANN [17]	40	0.008	**160**
Pomares 2000	3 × 5 (TP)	0.278	**23**
	4 × 6 (TP)	0.104	**39**
	5 × 9 (TP)	0.041	**72**
González 2001	5	0.3622 ± 0.0268	**20**
	10	0.1343 ± 0.0261	**40**

	15	0.0459 ± 0.0096	**60**
	21	0.0200 ± 0.0054	**84**
	29	0.0143 ± 0.0045	**116**
Rivas 2003	5	0.3666 ± 0.0168	**20**
	10	0.1108 ± 0.0135	**40**
	15	0.0368 ± 0.0092	**60**
	21	0.0191 ± 0.0036	**84**
	29	0.0147 ± 0.0022	**116**
PP-RBFNN	{1 4}	0.489 ± 0.0110	**15**
	{1 5}	0.365 ± 0.0006	**18**
	{1 6}	0.352 ± 0.0004	**21**
	{2 7}	0.128 ± 0.0021	**27**
	{3 7}	0.040 ± 0.0003	**30**
	{3 8}	0.026 ± 0.0015	**33**
	{4 8}	0.013 ± 0.0005	**36**
	{4 9}	0.007 ± 0.0022	**39**

Table.3 Comparative of different algorithms for the function $f_2(x)$

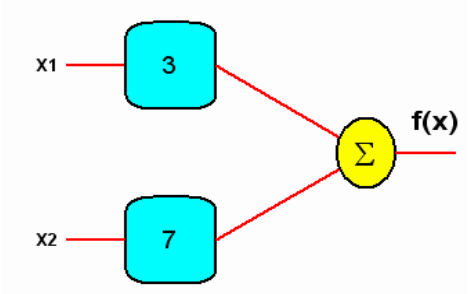


Fig. 10. Obtained hierarchical PP-RBFNN structure for the example $f_2(x)$

7 Conclusions

A fundamental limitation of the problem of approximation systems is that when the number of input variables increases, the number of parameters usually increases in a very rapid way, even

exponentially. This phenomenon prevents the use of the majority of conventional modelling techniques and forces us to look for more specific solutions. To deal with this problem, we proposed new architecture for modelling complex systems in function approximation problems. This architecture is composed of complete RBFNN that are in charge of a reduced set of input variables. Also we proposed a new method to select the more important input variables, thus reducing the dimension of the input variable space for each RBFNN. The selection of the hierarchical structure of PP-RBFNN adapted according to the selected number of input variables and which of these variables go alone or together in each RBFNN. We have also provided a method to find automatically the most suitable topology of the proposed hierarchical structure and a method to select the more important input variables. We showed that the results of PP-RBFNN outperform traditional methods in: number of parameters; number RBF and the approximation error.

References

1. Vehtari, A., Lampinen, J.: Bayesian Input Variable Selection Using Posterior Probabilities and Expected Utilities. Helsinki University of Technology, Laboratory of Computational Engineering publications. Report B. ISSN 1457-1404, 2002.
2. Kwasny, S. C., Kalman. B. L., Chang, N.: Distributed Patterns as Hierarchical Structures, World Congress on Neural Networks-Portland, OR. 1993.
3. Pomares, H., Rojas, I., González, J., Prieto, A.: Structure Identification in Complete Rule-Based Fuzzy Systems. IEEE Trans. On fuzzy systems, vol. 10, No. 3, June 2002
4. Chen, Y., Wang, J. Z.: Kernel machines and additive fuzzy systems: classification and function approximation. 789-795 vol.2. 2003
5. Bengio, S., Bengio, Y.: Taking on the curse of dimensionality in joint distributions using neural networks. IEEE Trans. Neur. Net., special issue on data mining and knowledge discovery, 11(3):550-557, 2000.
6. Gonzalez, J., Rojas, H., Ortega, J., Prieto, A.: A new clustering technique for function approximation. Neural Networks, IEEE Trans. on, Volume: 13 Issue: 1, Jan. 2002. Page(s): 132 - 142.
7. Fukumizu, K., Amari, S-I.: Local Minima and Plateaus in Hierarchical Structures of Multilayer Perceptrons. Brain Science Institute. The Institute of Physical and Chemical Research (RIKEN) October 22, 1999.
8. de Souza, F.J., Vellasco, M.M.R., Pacheco, M.A.C.: Hierarchical neuro-fuzzy quadtree models. Fuzzy Sets and Systems 130 (2002) 189–205.
9. Ferrari, S., Maggioni, M., Borghese, N.A.: Multiscale approximation with hierarchical radial basis functions networks. IEEE Trans. Neur. Net., vol.15, no.1, pp.178-188, 2004.
10. Moody J. and Darken C. Fast learning in networks of locally tuned units. Neural Computations.1989. 1(2):281-294.
11. S.Bengio and Y.Bengio. “Taking on the Curse of Dimensionality in Joint Distributions Using Neural Networks” IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 3, MAY 2000.
12. A.K. Jain and R .Chandrasekaran. “Dimensionality and sample size consideration in pattern recognition practice”. In P.R. Krishaniah and L.N. Kanal, editors, Handbook of Statistics, volume II, pages 835-855. North-Holland, Amsterdam, The Netherlands, 1982.

13. M.Awad, H.Pomares, F.Rojas, L.J. Herrera, J.González, A. Guillén. “Approximating I/O data using Radial Basis Functions:A new clustering-based approach”. IWANN 2005, LNCS 3512, pp. 289-296, 2005.© Springer-Verlag Berlin Heidelberg 2005.
14. J.H. Friedman. and W. Stuetzle, "Projection pursuit regression", Journal of the American Statistics Association, Vol. 76, No. 376, pp. 817-823, 1981.
15. J.H. Friedman, Multivariate Adaptive Regression Splines, Annals of Statistics, Vol 19, 1-141, 1991.
16. V. Cherkassky. and H. Lay-Najafy., "Constrained Topological Mapping for Nonparametric Regression Analysis", Neural Networks, vol. 4, pp. 2740,1991.
17. V. Cherkassky, D. Gehring, and F. Mulier, Comparison of adaptive methods for function estimation from samples, IEEE Trans. NN 7, 969-984,1996.
18. H.Pomares. “Nueva metodología para el diseño automático de sistemas difusos”. Tesis Doctoral, universidad de granada, 2000.
19. J. Gonzalez. “Identificación y Optimización de redes de Funciones de Base Radiales Para Aproximación funcional”. Tesis Doctoral, Universidad de Granada. 2001.
20. V.Rivas. “Optimización de Redes Neuronales de Funciones Base Radiales Mediante Algoritmos Evolutivos”. Tesis Doctoral, universidad de Granada 2003.

Article received: 2007-05-18