Proceedings of the 8th WSEAS International Conference on APPLIED MATHEMATICS, Tenerife, Spain, December 16-18, 2005 (pp228-233)

Hierarchical Structure for function approximation using Radial Basis Function

M.Awad, H.Pomares, I.Rojas, L.J.Herrera, A.Guillen, O.Valenzuela
Department of Computer Architecture and Computer Technology
E.T.S. Ingeniería Informática. University of Granada
Spain (irojas@atc.ugr.es)

Abstract: - The hierarchical network proposed (Multi-RBFNN), is composed of complete Radial Basis Function Neural Networks (RBFNNs) that are in charge of a reduced set of input variables with the property of which every Sub-RBFNN can take charge of a set of input variables and not of all. For the optimization of the whole net, 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 a Sub-RBFNN to build the hierarchic structure Multi-RBFNN, 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 hierarchical structure and selects the more important input variables for it.

Key-words: Hierarchical architectures, Input variable selection, Function approximation, Radial basis functions.

1. Introduction

RBFNNs is universal approximations, it corresponds to a particular class of function approximation that can be trained using a set of samples of I/O. On the other hand, when the structure of a RBFNN is complex (not hierarchical), a direct consequence is the production of a big network, with a big number of hidden units that it produce a big quantity of computational parameters and this impedes the convergence of the learning process and increases the requirements of memory and time. An effective solution is to incorporate a hierarchical structure adapted to solve problems of complex models. The ideas of modulating neural networks are essential for hierarchical designs [1][2][3][4]. In problems of function approximation, when increases the number of input variables, the number of parameters increases exponentially. A direct consequence is to have a huge network, with big number of neurons in the hidden layer that hinders the convergence of the process of training of the network. The hierarchical structure of radial basis function networks proposed (Multi-RBFNN) divides at first the problem of the function approximation in smaller problems based on the more important input variables that have been selected and which of these selected variables will be go alone or together in a Sub-RBFNN. The final structure is a hierarchical series of RBFNNs connected in parallel with an total output that is the linear sum of all the outputs of Sub-RBFNNs. This topology facilitates to solve problems that can be combined to gain access to a complex solution. Once the number of Sub-RBFNNs is known, the hierarchical structure Multi-RBFNN is constructed. In every Sub-RBFNN, which presents a radial basis function network (RBFNN), they optimized the parameters of the RBFNN (centres, radios, weight); using an efficient algorithm of clustering to initialize the value of the centres in every Sub-RBFNN. To optimize the values of the radius in every Sub-RBFNN, we used traditional algorithms. When the parameters of centres and radius of every Sub-RBFNN have been initialized, a method of linear calculation is used to find the exact values of the weight in the whole hierarchical system that minimizes the cost function calculated on the set of samples of I/O data.

2. The Architecture Of The Multi-RBFNN System

In RBFNNs every neuron in the hidden layer receives all the input variables of the network. Nevertheless, the interconnections in the hierarchical structure Multi-RBFNN between input variables and the hidden layer are limited and located. The advantage of the hierarchical structure Multi-RBFNN consists of the fact that the problem divides into many problems that connected in parallel. Every problem presents a RBFNN named Sub-RBFNN. All the Sub-RBFNN has a total output that is the output of the hierarchical structure Multi-RBFNN. This division of the system Multi-RBFNN limits the quantity of the information of the previous layer. In general, to construct a hierarchic structure Multi-RBFNN to solve problems of function approximation consists of two basic steps:

- The identification of the structure (input variables selection, distribution of the selected input variables to the number Sub-RBFNNs, the number of Sub-RBFNNs depends on the number of the selected input variables and on which of these variables go alone or together in a Sub-RBFNN).

- The estimation of the parameters of every Sub-RBFNN (centres $\vec{c}^{\,S}$, radius r^{s} and weight w^{S}, and the number of radial functions RBF in each Sub-RBFNN), and the calculation of the total output *f(x)* of the hierarchical structure Multi-RBFNN.

Fig. 1 presents the principal steps of the proposed algorithm. Fig. 2 presents the proposed hierarchical Multi-RBFNN system. Each one of the nodes of Fig. 2 is a Radial Basis Functions Network (see Fig. 3). RBFNNs can be seen as a particular class of Artificial Neural Networks ANNs. They are characterized by a transfer function in the hidden unit layer having radial symmetry with respect to a centre. The basic architecture of an RBFNN is a 3-layer network. The output of the net is given by the following expression: The output of the net is given by the following expression:

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

where $\Phi=\{\phi_i : i=1,\dots,m\}$ are the basis functions set and w_i the associate weights for

every RBF. The basis function ϕ can be calculated as a gaussian function using the following expression:

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

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

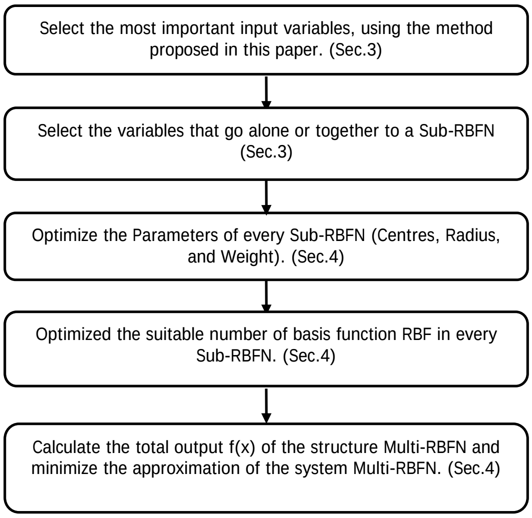


Fig. 1. Principal steps of the proposed algorithm

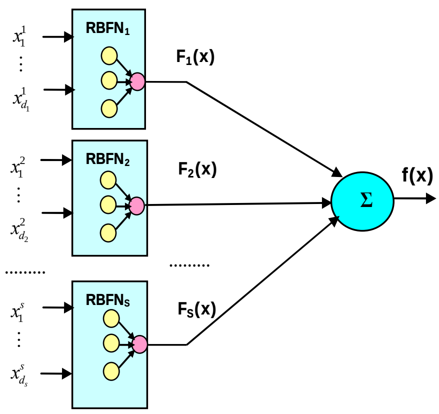


Fig. 2. The hierarchical structure Multi-RBFNN.

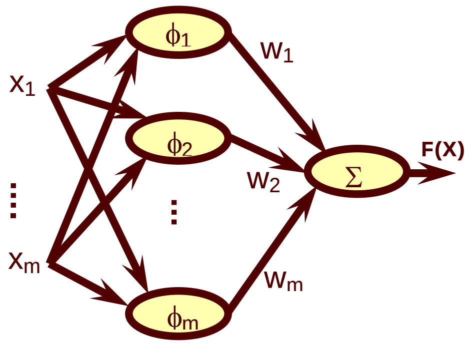


Fig. 3. Each of the sub-networks is a RBFNN

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

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

Where $\Phi_i^s=\{\phi_i^s : i=1,\dots,n,\ S=1,\dots,s\}$ is the of activation matrix of the set of basis functions in all Sub-RBFNNs, w_i^s the associate weight of all Sub-RBFNN and S is the number of Sub-RBFNN. The process of the linear optimization of the weight depends to the activation matrix of the total output of the Multi-RBFNN $f(x)$. This process uses the direct methods as the singular value decomposition (SVD) to calculate the values of the weight w_i^s.

The proposed hierarchical structure Multi-RBFNN decreases the number of parameters which decrease the complexity of the approximate system and increases the efficiency of the process of function approximation from a set of examples of I/O data data.

3. Input Variables Selection For The Structure Multi-RBFNN

The input variables selection (IVS) tries to reduce the dimension of variables of input space and create a new set of input variables. This process of identification and elimination of so much irrelevant and redundant information as they are possible, reduces the dimensionality of the date set and allows algorithms of learning works more rapid and effectively.

The curse of the dimensionality [5] 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 have to cover or represent each part of his input space in order to know how that part of the input space should be mapped. Covering the input space takes 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 [6].The curse of the dimensionality cause networks with many irrelevant inputs that behave relatively badly, when the dimension of the input space is high the network uses almost all his resources to represent irrelevant parts of the input space. Even if we have a network algorithm which is able to focus on important portions of the input space, the higher the dimensionality of the input space, the more data may be needed to find what is important and what is not. A priori information can help with the curse of dimensionality. Input variable selection fundamentally affects the severity of the problem, as well as the selection of the neural network model. [7].

In this section we propose a new method for input variables selection for the problem of function approximation, and more specifically for our Multi-RBFNN system. This method considers a simple calculation to select the input variables. The selection of the input variables will do by the following steps:

1) Relate each possible input dimension of data $\{x_1,\dots,x_d\}$ with the dependent variable y (as a function in one dimension), as in the following expression:

$$\{(x_1,y),(x_2,y),(x_3,y),\dots,(x_d,y)\}$$

2) Divide the date in each dimension in P parts (the number of the parts depends on the number of the input data n, y when the number of the input data is big, and the number of parts P must increase). This division is obtained by means of the following expression:

$$\left\{P_i^{j-1} \le \left(\vec{x}^{\,k}\right)_i < P_i^j\right\} \quad k = 1,\dots,n; i = 1,\dots,d;\ j = 1,\dots,p$$

where n is the number of data of I/O, $\left(\vec{x}^{\,k}\right)_i$ it is the component i_{th} of the input vector k_{th}.

3) Associate the data of each dimension to corresponding output data as in the following expression:

$$\left\{\left(\vec{x}^{\,k}\right)_i, y^k\right\} \Big/ P_i^{j-1} \le \left(\vec{x}^{\,k}\right)_i < P_i^j$$

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:

$$D_i^j = \max(y_j^k) - \min(y_j^k) \quad j = 1,\dots.p$$

5) Finally, for each input variable x_i we calculate the mean of distances $\bar{D}_i$. The smaller $\bar{D}_i$, is the most important input variable for the problem, since this implies that the other variables affect very little to the output variable for every fixed value (partition interval) of x_i. Fig. 4 presents, in a schematic way, the general description of the proposed IVS method. For all the parts the average of the distance is calculated $\bar{D}$.

4. SIMULATION EXAMPLES

In this section different examples will be appear to verify the procedure in the proposed algorithm. Two types of results will be present: the structure of the system Multi-RBFNN selected by the algorithm using the method of IVS and which of the input variables must go alone or together a Sub-RBFNN in the system Multi-RBFNN, and the results of the validity of algorithm in approximate functions from samples of information 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. This way, the system Multi-RBFNN will be evaluated with his characteristics in decreases the number of parameters, which obtains the principal objective in the present work of the search of new architectures of calculation capable of shaping complex systems of function approximation, without the increase of the number of input variables has to suppose an exponential increase in the complexity of the system.

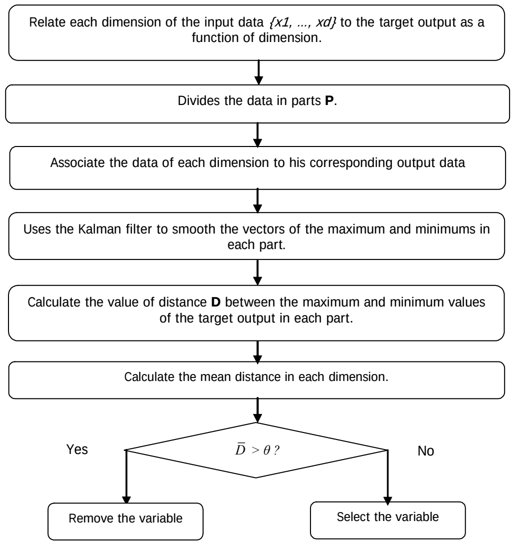


Fig. 4. General description of the method IVS.

The presents results of 5 executions; the set of radial functions used in the Sub-RBFNN **{RBF}** that there is considering the algorithm (every time there is added 1 RBF), **# Param** is the number of parameters. $\mathbf{NRMSE_{Tr}}$ is the normalized mean squared error of training and $\mathbf{NRMSE_{Test}}$ is the normalized mean squared error of test.

4.1. First Example $f_1(x)$

We will 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.

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

where each input variable is defined in the interval [0,1]. The proposed algorithm selects the ideal architecture of the system Multi-RBFNN for the function $f_1(x)$, depends to the value of the variance threshold after analyzing every variables (Fig. 5).

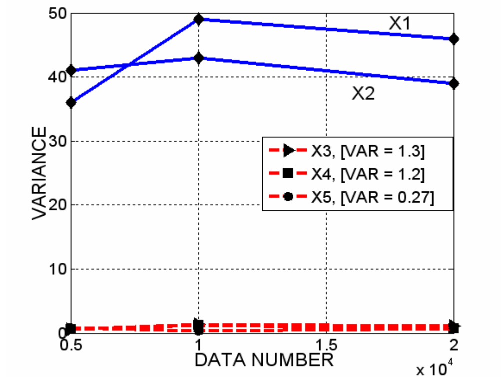
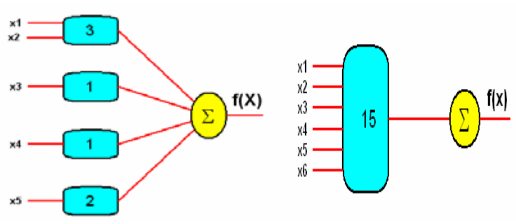


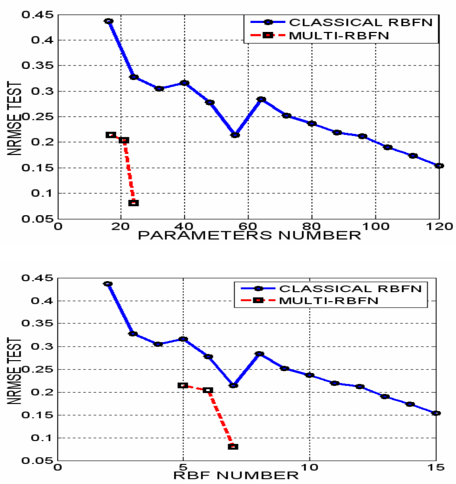
Fig. 5. The variance for each variable in $f_1(x)$

In the function $f_1(x)$ few variables must go alone to Sub-RBFNN and the subset of the rest goes to Sub-RBFNN, as in the Fig. 6.



(a) **(b)**

Fig. 6. (a) Structure Multi-RBFNN selected by the algorithm. (b) Structure of a classic RBFNN for the current function



(a)

(b)

Fig. 7. Comparison result between Multi-RBFNN system and Classical RBFNN. (a) In the number of parameters. (b) In the number of RBF.

Multi-RBFNN algorithm						Classical RBFNN			
{ RBF }	# Param	$NRMSE_{Tr}$	Std	$NRMSE_{Test}$	Std	RBF	# Param	$NRMSE_{Tr}$	$NRMSE_{Test}$
{2 1 1 1}	17	0.212	2E-3	0.214	1E-4	**2**	16	0.428	0.437
{1 2 1 1}	16	0.246	6E-3	0.252	4E-4	**3**	24	0.331	0.328
{1 1 2 1}	16	0.238	1E-2	0.243	5E-3	**4**	32	0.301	0.305
{1 1 1 2}	16	0.241	1E-2	0.246	6E-4	**5**	40	0.316	0.316
{3 1 1 1}	24	0.198	2E-1	0.204	1E-4	**6**	48	0.279	0.278
{2 2 1 1}	18	0.221	9E-3	0.225	1E-2	**7**	56	0.213	0.214
{2 1 2 1}	18	0.209	2E-3	0.216	6E-3	**8**	64	0.284	0.284
{2 1 1 2}	18	0.212	1E-3	0.216	1E-4	**9**	72	0.249	0.252
{4 1 1 1}	33	0.183	1E-2	0.189	8E-3	**10**	80	0.231	0.237
{3 2 1 1}	25	0.146	8E-2	0.147	3E-2	**11**	88	0.211	0.219
{3 1 2 1}	25	0.075	5E-3	0.084	3E-3	**12**	96	0.206	0.212
{3 1 1 2}	25	0.080	3E-3	0.088	2E-3	**13**	104	0.179	0.190
						14	112	0.153	0.173
						15	120	0.144	0.154

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

4.2 Second Example $f_2(x)$

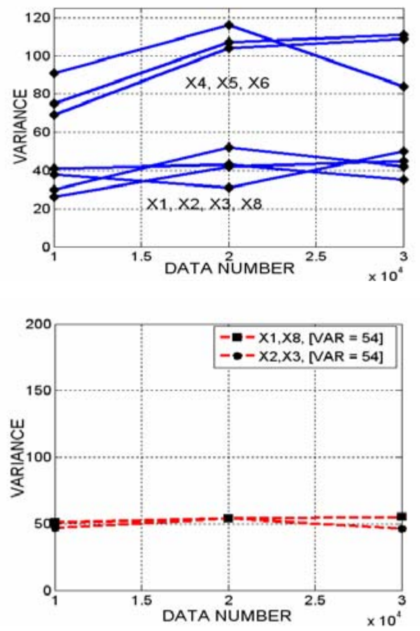
The training and test sets have been formed by 25000 randomly points.

$$f_2(x)=2\cos(2\pi x_1\cdot x_8)+6e^{(-2x_2\cdot x_2)}+7.5(x_4\cdot x_5\cdot x_6)+0x_7$$

$$,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8\in[0,1]$$

The algorithm selects the ideal architecture of the system Multi-RBFNN, depends to the value of the variance threshold after analyzing every variables (Fig. 8).

(a)



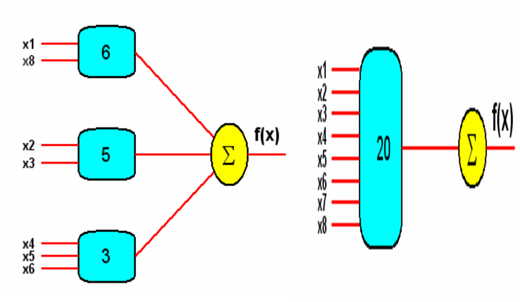
(b)

Fig. 8. The variance in different data number. (a) For each variable. (b) For each subset of two variables

In the function $f_2(x)$ tow subset of tow variables go to Sub-RBFNN and the subset three variables go to Sub-RBFNN, as in the Fig. 9.

Multi-RBFNN algorithm						Classical RBFNN			
{ RBF }	# Param	$NRMSE_{Tr}$	Std	$NRMSE_{Test}$	Std	RBF	# Param	$NRMSE_{Tr}$	$NRMSE_{Test}$
{2 1 1}	17	0.422	1E-3	0.426	3E-3	2	20	0.459	0.473
{1 2 1}	17	0.451	1E-2	0.457	1E-2	3	30	0.433	0.450
{1 1 2}	18	0.498	6E-3	0.503	9E-3	4	40	0.382	0.393
{3 1 1}	21	0.419	2E-3	0.424	6E-4	5	50	0.397	0.416
{2 2 1}	21	0.260	4E-3	0.268	7E-3	6	60	0.380	0.402
{2 1 2}	22	0.423	6E-4	0.427	2E-3	7	70	0.339	0.354
{3 2 1}	25	0.269	4E-3	0.277	7E-3	8	80	0.349	0.372
{2 3 1}	25	0.261	6E-3	0.269	4E-2	9	90	0.344	0.366
{2 2 2}	26	0.307	3E-2	0.314	3E-2	10	100	0.316	0.341
{3 3 1}	29	0.247	6E-2	0.254	6E-2	11	110	0.328	0.351
{2 4 1}	29	0.264	7E-3	0.273	5E-2	12	120	0.259	0.284
{2 3 2}	30	0.260	5E-2	0.269	4E-3	13	130	0.250	0.262
{3 3 2}	32	0.225	8E-2	0.233	8E-3	14	140	0.241	0.249
{2 4 2}	34	0.261	3E-2	0.269	3E-2	15	150	0.234	0.241
{2 3 3}	35	0.233	5E-2	0.241	5E-2	16	160	0.216	0.231
{3 3 3}	39	0.221	8E-2	0.229	8E-2	17	170	0.208	0.215
{2 4 3}	39	0.207	8E-2	0.215	8E-2	18	180	0.189	0.190
{2 3 4}	40	0.214	8E-2	0.222	6E-2	19	190	0.177	0.185

Table III NRMSE of training and test obtained by the proposed algorithm and by classic RBFNN for the function $f_2(x)$



(a) **(b)**

Fig. 9. (a) Structure Multi-RBFNN selected by the algorithm. (b) Structure of a classic RBFNN for the current function

5. CONCLUSIONS

A fundamental limitation in 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 have searched for new architectures for modelling complex systems in function approximation problems. The new hierarchical network proposed is composed of complete Radial Basis Function Networks that are in charge of a reduced set of input variables. For this architecture, we have 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 Multi-RBFNN adapted according to the selected number of input variables and which of these variables go alone or together in a Sub-RBFNN.

Acknowledgement
This work has been partially supported by the Projects CICYT TIC2002-11350-52E and TIN2004-01419.

References:

[1] 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.

[2] 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.

[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] S-Y.Taur.Kung,Sh-H.Lin. “Hierarchical Fuzzy Neural Networks for Pattern Classification”. National Science Council through Grant NSC 88-2213-E-005-012. 2002.

[5] R.Bellman. “Adaptive Control Processes: A Guided Tour”. Princeton University Press. 1961.

[6] 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.

[7] 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.