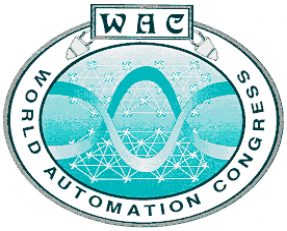


World Automation Congress

Fifth International Symposium on Intelligent Automation and Control

Seville, Spain
June 28th-July 1st, 2004

B-Spline And Neuro-Fuzzy Models Design With Function And Derivative Equalities

C. Cabrita and A. Ruano

B-SPLINE AND NEURO-FUZZY MODELS DESIGN WITH FUNCTION AND DERIVATIVE EQUALITIES

C. Cabrita, Centre for Intelligent Systems, Univ. Algarve, Portugal, ccabrita@ualg.pt

A. Ruano, Centre for Intelligent Systems, Univ. Algarve, Portugal, aruano@ualg.pt

Abstract: The design of neuro-fuzzy models is still a complex problem, as it involves not only the determination of the model parameters, but also its structure. Of special importance is the incorporation of a priori information in the design process. In this paper two known design algorithms for B-spline models will be updated to account for function and derivatives equality restrictions, which are important when the neural model is used for performing single or multi-objective optimization on-line.

KEYWORDS: constructive algorithms, B-Splines, neuro-fuzzy systems, genetic programming;

1. INTRODUCTION

Neural network and neuro-fuzzy models are nowadays a valid alternative to classical models used in control systems, such as ARX and ARMAX models. However, the design of the structure of such models for the particular problem at hand is still a complex problem, with several heuristics being proposed for each type of neural and neuro-fuzzy model. Of special importance is the incorporation of a priori knowledge into the design process, and this is related to the ultimate goal of the model within the control structure. This paper focuses on the use of models for performing single or multi-objective optimization on-line. If the models are to be updated on-line, then models that store information locally should be used. For this reason, B-spline models are employed here, and, thanks to their functional equivalence, directly applicable to Mamdani fuzzy models (satisfying some assumptions) and to Takagi-Kang-Sugeno (or simply Takagi-Sugeno) models [1]. For this type of networks there is a known design heuristic, the ASMOD algorithm [2]. More recently, the authors have proposed the use of genetic programming [3] as a valid alternative for the design of such a type of models. In the present paper, both algorithms will be updated for the specific use described above. If the user has an a priori knowledge of the location of the minima of the function to be approximated, then this knowledge can be incorporated in the design procedure, and assured that the resulting network satisfies user-defined function and derivative equalities.

2. B-SPLINE NEURAL NETWORKS

B-spline neural networks belong to the class of networks termed *grid or lattice-based associative memories networks* (AMN). This type of networks is composed of three layers: a *normalised input space* layer, a *basis functions* layer and a *linear weight* layer.

The normalised input space layer is usually a grid on which the basis functions are defined. In order to define a grid in the input space, vectors of *knots* must be defined, one for each input axis. They are arranged in such a way that:

$$x_i^{\min} < \lambda_{i,1} \le \lambda_{i,2} \le \cdots \le \lambda_{i,r_i} < x_i^{\max}, \tag{1}$$

where $x_i^{\min}$ and $x_i^{\max}$ are the minimum and maximum values of the i^{th} input, respectively.

The j^{th} univariate basis function of order k is denoted $N_k^j(x)$, and it is defined by the following relationships:

$$N_k^j(x) = \left(\frac{x-\lambda_{j-k}}{\lambda_{j-1}-\lambda_{j-k}}\right) N_{k-1}^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-k+1}}\right) N_{k-1}^j(x),\ N_1^j(x) = \begin{cases} 1 \ \textit{if} \quad x \in I_j \\ 0 \ \textit{otherwise} \end{cases} \tag{2}$$

The output of the hidden layer is determined by a set of p basis functions defined on the n-dimensional grid. The shape, size and distribution of the basis functions are characteristics of the particular AMN employed, and the support of each basis function is bounded.

The output of an AMN is a linear combination of the outputs of the basis functions. The linear coefficients are the adjustable weights, and as the mapping is linear, finding the weights is just a linear optimization problem. The output is therefore:

$$y = \sum_{i=1}^{p} \mathbf{a}_i \mathbf{w}_i = \mathbf{a}^T \mathbf{w}\,, \tag{3}$$

where $\mathbf{a}_i = N_{\mathbf{k}}^i(x), \quad i = 1,\ldots,p$. As only $p'' = \prod_{i=1}^{n} \mathbf{k}_i$ are active at any one time, the calculation of (3) can be reduced to:

$$y = \sum_{i=1}^{p''} \mathbf{a}_{act(i)}(\mathbf{x})\, \mathbf{w}_{act(i)}\,, \tag{4}$$

where $\mathbf{a}_{act(i)}(\mathbf{x})$ denotes the i^{th} active basis function for input $\mathbf{x}$.

To overcome the "curse of dimensionality", it is common to employ, instead of a single module covering all inputs, a linear sum of smaller sub-modules, each one with a lower input dimensionality. The output of such a network is:

$$y(\mathbf{x}) = \sum_{u=1}^{n_u} S_u\left(\mathbf{x}_{\underline{u}}\right) \tag{5}$$

where $S_i\left(\mathbf{x}_{\underline{i}}\right)$ denotes the i^{th} sub-model, and $\mathbf{x}_{\underline{i}}$ is the set of input variables ($\underline{i}$) which compose sub-model i.

3. INCORPORATING EQUALITY RESTRICTIONS

Assuming that the network structure has been already discovered, the final design stage is to determine the knot placement and the linear output vector. Denoting as $\boldsymbol{\lambda}$ the vector of all the interior knots, and as $\mathbf{z}$ the vector of the design parameters:

$$\mathbf{z} = \begin{bmatrix} \mathbf{w} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{6}$$

In order to exploit the topology of the neural network by the training algorithm, and denoting the basis function output matrix as $\mathbf{A}$, then the network output vector is just:

$$\mathbf{y} = \mathbf{A}(\boldsymbol{\lambda})\mathbf{w} \tag{7}$$

For any combination of the interior knots, the optimal value of the linear parameters is:

$$\hat{\mathbf{w}} = \mathbf{A}^{+}(\boldsymbol{\lambda})\mathbf{t} \tag{8}$$

Let us assume that, together with the minimization of (8), we want to add some equality restrictions such as:

$$\mathbf{y}(\mathbf{X}) = \underline{\mathbf{y}} \tag{9}$$

or

$$\frac{\partial \mathbf{y}(\mathbf{X})}{\partial \mathbf{x}_i} = \underline{\mathbf{y}}'(\mathbf{X}_i)\,, \tag{10}$$

where $\boldsymbol{X}$ denotes the m input points where the equalities should hold. Equalities of both types can be recast as a linear system of equations:

$$\mathbf{R}\mathbf{w}_d = \mathbf{b} \tag{11}$$

Where rank($\mathbf{R}$)<k, k being the number of columns of $\mathbf{A}$.

Let us assume that m restrictions should be satisfied. Typically, only a subset of all the linear weights will be active for these restrictions. Let these weights be denoted as dependent ($\mathbf{w}_d$) on the values of the other ($\mathbf{w}_i$) weights. Assuming, without loss of generality, that the dependent ones will be the first weights, (8) can be recast as:

$$\begin{bmatrix} \hat{\mathbf{w}}_{d_{constr}} \\ \hat{\mathbf{w}}_{i_{unconstr}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{w}}_{d_{unconstr}} - \mathbf{H}^{-1}\mathbf{R}^T\left(\mathbf{R}\mathbf{H}^{-1}\mathbf{R}^T\right)^{-1}\left(\mathbf{R}\hat{\mathbf{w}}_{d_{unconstr}} - \mathbf{b}\right) \\ \hat{\mathbf{w}}_{i_{unconstr}} \end{bmatrix}, \tag{12}$$

where $\hat{\mathbf{w}}_{unconstr}$ is given by (8), and $\mathbf{H}^{-1} = \mathbf{A}_d{}^T\mathbf{A}_d$, $\mathbf{A}_d$ being the column partition related with the dependent weights.
The computation of **R** is straightforward for the case of function restrictions. It is slightly more complicated for the case of derivative restrictions.

3.1. Derivative Restrictions

Considering n_u sub-modules, we shall denote by S_{u^i} those sub-modules that depend on the input variable *x*, and by S_u those that do not depend on x. Therefore, performing the analysis for each individual equality:

$$y(\mathbf{z}) = \sum_{u=1}^{n_{u^i}} S_{u^i}(\mathbf{X}_{\underline{u}}) + \sum_{u=1}^{n_u} S_u(\mathbf{X}_{\underline{u}}) \tag{13}$$

Obviously, the derivative of the 2nd sum is null. Therefore:

$$\frac{\partial y(\mathbf{z})}{\partial x} = \sum_{u=1}^{n_{u^i}} \frac{\partial S_{u^i}(\mathbf{X}_{\underline{u}})}{\partial x} \tag{14}$$

Considering now the derivative of each sub-model, we assume, for the sake of simplicity, that only 2 input variables (x and y) intervene in this sub-model. We have:

$$\frac{\partial S_{u^i}(\mathbf{X}_{\underline{u}})}{\partial x} = \frac{\partial}{\partial x}\sum_{i=1}^{r_x+k_x}\sum_{j=1}^{r_y+k_y} \mathbf{w}_{i,j} N_{k_y}^j(y) N_{k_x}^i(x) = \sum_{j=1}^{r_y+k_y} N_{k_y}^j(y) \frac{\partial}{\partial x}\sum_{i=1}^{r_x+k_x} \mathbf{w}_{i,j} N_{k_x}^i(x) \tag{15}$$

Using only the non-zero derivatives, we have:

$$\frac{\partial S_{u^i}(\mathbf{X}_{\underline{u}})}{\partial x} = \sum_{j=1}^{r_y+k_y} N_{k_y}^j(y) \sum_{i=1}^{r_x+k_x-1} (k_x - 1)\frac{\mathbf{w}_{i+1,j} - \mathbf{w}_{i,j}}{\lambda_{x_i} - \lambda_{x_{i-k_i+1}}} N_{k_x-1}^i(x) \tag{16}$$

Therefore, the derivative of a multidimensional sub-model is another multidimensional sub-model, with the same number of uni-dimensional sub-models, all with the same order, except the one related with the variable for which the derivative is taken, which has its order decreased by one unit. All the weights are updated, according to:

$$\mathbf{w}_{i,j} = (k_x - 1)\frac{\mathbf{w}_{i+1,j} - \mathbf{w}_{i,j}}{\lambda_{x_i} - \lambda_{x_{i-k_i+1}}} \tag{17}$$

In order to satisfy the restrictions, eq. (16) can be expressed in another form.

$$\frac{\partial S_{u^i}(\mathbf{X}_{\underline{u}})}{\partial x} = \sum_{j=1}^{r_y+k_y} N_{k_y}^j(y)\left(\begin{array}{l} \sum_{i=1}^{r_x+k_x-1} \frac{(k_i-1)}{\lambda_{x_i} - \lambda_{x_{i-k_x+1}}} N_{k_x-1}^i(x)\mathbf{w}_{i+1,j} - \\ -\sum_{i=1}^{r_x+k_x-1} \frac{(k_x-1)}{\lambda_{x_i} - \lambda_{x_{i-k_x+1}}} N_{k_x-1}^i(x)\mathbf{w}_{i,j} \end{array}\right) = \sum_{j=1}^{r_y+k_y} N_{k_y}^j(y)\left(\sum_{i=1}^{r_x+k_x-1} a_i\mathbf{w}_{i+1,j} - \sum_{i=1}^{r_x+k_x-1} a_i\mathbf{w}_{i,j}\right)$$

$$= \sum_{j=1}^{r_y+k_y} N_{k_y}^j(y)\left(-a_1\mathbf{w}_{1,j} + \sum_{i=2}^{r_x+k_x-1}(a_{i-1} - a_i)\mathbf{w}_{i,j} a_{r_x+k_x-1}\mathbf{w}_{r_x+k_x-1,j}\right) \tag{18}$$

where:

$$a_i = \frac{(k_i-1)}{\lambda_{x_i} - \lambda_{x_{i-k_x+1}}} N_{k_x-1}^i(x), \quad i = 1\cdots r_x + k_x - 1 \tag{19}$$

Note that the complexity of (18) is misleading. As we know, there are only k_i active splines in the i[th] dimension. Therefore, for any one point, only $k_x k_y$ weights need to be considered, or in the case of more than 2 input variables, $\prod_{i=1}^{n} k_i$ weights are non-null.

To ensure (10), for just one restriction, we need to solve just 1 equation. Consider that our point lies in the intervals I_{x_i} and I_{y_i}. The active basis functions are $i \in \left[i_s, i_{s+k_x} \right[$ and $j \in \left[j_s, j_{s+k_x} \right[$ for the inputs *x* and *y* considered.

$$\underline{y}'_x(\mathbf{z}) = \frac{\partial S_{u^i}}{\partial x} = \sum_{j=j_s}^{j_s+k_y-1} N_{k_y}^{j}(y) \left(\begin{array}{l} -a_{i_s} \mathbf{w}_{i_s,j} + \sum_{i=i_s+1}^{i_s+k_x-1} (a_{i-1} - a_i) \mathbf{w}_{i,j} + \\ +a_{i_s+k_x-1} \mathbf{w}_{i_s+k_x,j} \end{array} \right) \tag{20}$$

Therefore the dependent weight can be any of $\mathbf{w}_{i,j}$, within those limits. Assuming, without loss of generality that the dependent weight is the first one, $\mathbf{w}_{i_s,j_s}$, it can be given as:

$$\mathbf{w}_{i_s,j_s} = \frac{\underline{y}'_x(\mathbf{z}) - \sum_{j=j_s}^{j_s+k_y-1} N_{k_y}^{j}(y) \left(\begin{array}{l} -a_{i_s} \mathbf{w}_{i_s,j} + \sum_{i=i_s+1}^{i_s+k_x-1} (a_{i-1} - a_i) \hat{\mathbf{w}}_{i,j} \\ +a_{i_s+k_x-1} \hat{\mathbf{w}}_{i_s+k_x,j} \end{array} \right)}{N_{k_y}^{j_s}(y) a_1} - \frac{\sum_{i=i_s+1}^{i_s+k_x-2} (a_{i-1} - a_i) \hat{\mathbf{w}}_{i,j_s} + a_{i_s+k_x-1} \hat{\mathbf{w}}_{i_s+k_x,j_s}}{a_1} \tag{21}$$

If there are more sub-modules that depend on the variable x, (21) can be updated to:

$$\mathbf{w}_{i_s,j_s} = \frac{\underline{y}'_x(\mathbf{z}) - \sum_{j=j_s}^{j_s+k_y-1} N_{k_y}^{j}(y) \left(\begin{array}{l} -a_{i_s} \mathbf{w}_{i_s,j} + \sum_{i=i_s+1}^{i_s+k_x-1} (a_{i-1} - a_i) \hat{\mathbf{w}}_{i,j} + \\ +a_{i_s+k_x-1} \hat{\mathbf{w}}_{i_s+k_x,j} \end{array} \right)}{N_{k_y}^{j_s}(y) a_1} - \frac{\sum_{i=i_s+1}^{i_s+k_x-1} (a_{i-1} - a_i) \hat{\mathbf{w}}_{i,j_s} + a_{i_s+k_x-1} \hat{\mathbf{w}}_{i_s+k_x,j}}{a_{i_s}} - \frac{\sum_{z=2}^{n_{u^i}} \sum_{j=j_s}^{j_s+k_y} N_{k_y}^{j}(y) \left(\begin{array}{l} -a_1 \hat{\mathbf{w}}_{i_s,j} + \sum_{i=i_s+1}^{i_s+k_x-2} (a_{i-1} - a_i) \hat{\mathbf{w}}_{i,j} + \\ +a_{i_s+k_x-1} \hat{\mathbf{w}}_{i_s+k_x,j} \end{array} \right)}{N_{k_y}^{j_s}(y) a_1} \tag{22}$$

Usually, the most important restrictions are those that annul the gradient vector (totally or partially) which means that the r.h.s in (21) or (22) is null. Considering that there are *n* inputs, and the aim is to annul *k* out of these *n* derivatives, (20) can be recast in the form (11), where the dimension of **R** is *k***k*.

4. EVOLVING THE STRUCTURE

In the last section, the structure of the network is assumed fixed. In this section, the structure of the network is evolved, by modifying a well known algorithm, ASMOD [2], and a recently proposed alternative [3], using genetic algorithms. By lack of space, for a detailed description of these algorithms, the reader is referred to the references described. The updated algorithms, that guarantee that restrictions (9) and (10) are met, will be denoted as RBMOD and RBGEN, respectively. Common updates to both algorithms are described below. It should be pointed out that specific updates to each algorithm were required, but by lack of space, can not be described .

4.1. Initial Model Creation

In the presence of function or derivative restrictions, the initial model structure must satisfy some features which will be referred next.

- if there are function restrictions, all input variables must be present in the model;
- For the sake of simplicity, the first variable in the model supplies the necessary basis functions that satisfy the function restrictions;
- In case of existence of derivative restrictions, each sub-model's variable subject to this kind of restrictions must supply a sufficient amount of basis functions, probably requiring the addition of interior knots;
- A given variable subject to derivative restrictions, may not be subject to knots addition because there exists another sub-model with the same variable;
- The operation of adding knots requires that each extra knot be placed among (1 knot if the number to insert is smaller or equal to the number of restriction patterns in the variable, 2 knots otherwise) the restriction patterns. However, the number of extra knots needs not to be greater than the *spline_order* among each each pair of restriction patterns;
- An optimised model structure satisfies the number of restrictions as long as the number of input restriction patterns contained in two adjacent interior knots does not overcome the variable order;

The extra number of interior knots required is computed from:

$$N\_knots=N\_FunctionRestrictions+N\_DerivativeRestrictions-OrderOfSpline \quad (23)$$

4.2. Structure Pruning

Because this phase aims to simplify the model's structure, it may occur that not every restriction is fulfilled. Therefore, there are some considerations to be accounted for:

Variable order reduction.
Neither function nor derivative restrictions: the variable order may be reduced or the variable can be removed.
Derivative but no function restrictions or *Derivative and function restrictions*: the variable order can be reduced as long as its order is greater than 2.
Function but no derivative restrictions: the variable order may be freely reduced, but the variable can not be removed.

Splitting multi-variable sub-models.
This procedure is unchanged since the initial model consists of univariate sub-models only, corresponding to an adequate structure.

Reducing number of interior knots.
The restrictions are satisfied as long as the number of interior knots generates as many basis functions as the number of restrictions, however, no more than *OrderOfSpline* input patterns subject to restrictions can be among two adjacent knots, should derivative restrictions exist.

5. RESULTS

In order to show some graphical results, we have chosen a problem composed of 2 input variables. This function is defined as $f(x,y)=x^2y^2-xy$, which contains infinite global optimums for $f(x,y)=-\frac{1}{4}\Big|_{y=\frac{1}{2x}}$.

The input data was generated within the interval [-1,1], at samples of 0.125.

The restrictions were imposed on the patterns, $(x_0,y_0)=\{(-1,-0.5),(-0.5,-1),(0.5,1),(1,0.5)\}$. Both function and derivative restrictions were set on the former patterns, for every input variable, such that $\frac{\partial \mathbf{y}(\mathbf{X})}{\partial \mathbf{x}_i}=\underline{\mathbf{0}}$.

For RBGEN, the terminal type mutation rate is: [5%, 5%, 20%, 40%, 30%].

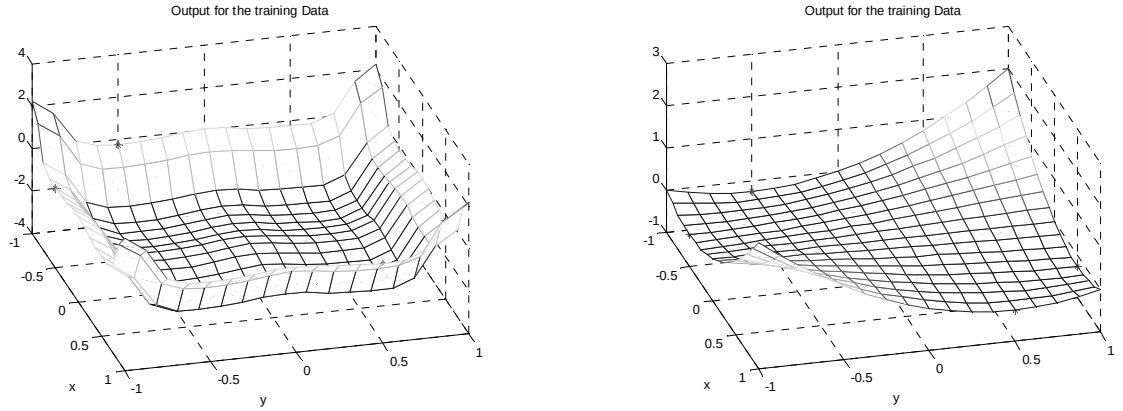
Table 1: Parameters definition for the RBGEN algorithm.

Parameters	N_ind	N_ger	Crossover Rate	Mutation Rate
N_ind	10	10	50%	0.8

We have run the two algorithms once, and the results are summarized in Table 2. We observe that function aproximation is almost ideal, with the RBGEN generating a model of lower complexity than the one achieved by RBMOD

Table 2: Final values for the best candidate.

Algorithm	BIC	‖w‖	MSE	Numb. Candidates	Model Complex.	Sub-Models
RBMOD	-17674	27.8	$3.3x10^{-28}$	110	108	1+2+[1x2]
RBGEN	-19595	11.3	$1.3x10^{-30}$	100	52	[1x2]



a) First iteration b) Last iteration

Figure 1: The output of the B-spline Network, for the test function for the best candidate, using RBMOD. The red '' points indicate the specified optima of the function.*

By observing *Figure 1* it can be verified how poor the initial solution is, in contrast to the final one, where the number of interior knots added provided a broader margin for data fitting, showing the correct function approximation and maintaining the restrictions required.

6. CONCLUSIONS

Two known algorithms have been updated to account for function and derivative restrictions. This enables, when neuro-fuzzy models are to be used for on-line optimization, to incorporate a priori information. Future work will address the application of these updated algorithms in practical applications.

REFERENCES

[1] Ruano, A.E., Cabrita, C., Oliveira, J.V., Kóczy, L.T., *Supervised Training Algorithms for B-Spline Neural Networks and Neuro-Fuzzy Systems*, International Journal of Systems Science, **33**, 8, 2002, pp. 689-711

[2] Weyer E., T. Kavli, *The ASMOD Algorithm. Some New Theoretical and Experimental Results*. SINTEF Report STF31 A95024, Oslo, 1995.

[3] C. Cabrita, A. E. Ruano, C.M. Fonseca, *Single and Multi-Objective Genetic Programming Design for B-Spline Neural Networks and Neuro-Fuzzy Systems*, IFAC Workshop on Advanced Fuzzy/Neural Control 2001, (AFNC'01), Valencia, Spain, 15-16, 2001, pp. 93-98