In Proceedings of the Fourth European Workshop on Fuzzy Decision Analysis and Recognition Technology, Dortmund, 1999

A Comparative Study of B-Spline Fuzzy Controller and RBFN

Jianwei Zhang, Wali Baqai and Alois Knoll
Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany
Phone: ++49-(0)521-106-2951 Fax: ++49-(0)521-106-2962
Email: zhang@techfak.uni-bielefeld.de

Abstract

Using of B-Spline functions as membership functions (MFs) in a fuzzy controller is an efficient method in data modelling and rule extraction. Als alternatives to the B-spline MFs, radial basis functions, e.g. Gaussian functions, are also often used in building neuronal models, which is called radial basis function network (RBFN). In this paper we present the results of comparing these two models by testing a numerous of analytical functions and checking how closely the trained models converge to the original functions. The B-spline fuzzy controller (B-FC) with equidistant distributed B-splines performs well in most cases, and B-FC optimised with genetic algorithm (GA) performs best in all cases.

Keywords

Radial Basis Function, B-spline, Genetic Algorithm, Function Approximation

1 Introduction

Fuzzy controller is an important model for abstraction complex data thanks to its interpretability and function approximating capability. However, the choice of MF affects how precise the fuzzy controllers are able to approximate functions. In [4] the common MFs like triangles, trapezoids and other set functions were employed and compared. In our previous work [5] we showed the advantages of using B-splines as MF. We compared splines and a fuzzy controller with SISO (*single-input-single-output*) and MISO (*multi-input-single-output*) structures. An alternative to the B-spline is the Gaussian function. Neuro fuzzy models with Gaussian functions as MF form the popular radial basis function network. This paper will first briefly introduce the principle of constructing RBFN and B-FC, then discuss the problem of automatical optimisation B-FC with genetic algorithm. The main purpose of this paper is to compare of B-FC with RBFN in relation to function approximation.

2 RBFN

2.1 The Popularity of RBFN

RBFNs were first introduced by Broomhead and Lowe in the paper titled *Multivariable function interpolation and adaptive networks* [1]. Also they proposed a procedure for designing a layered adaptive network which implements the method of RBFN. The use of the network for solving the XOR problem and prediction of chaotic time series were demonstrated. Although the basic idea of RBFN was developed 30 years ago under the name *method of potential function*, the paper by Broomhead and Lowe has opened a frontier in the neural network community. In 1989, exactly one year since the publishing of the paper of Broomhead and Lowe, Moody and Darken published their paper titled *Fast learning in network of locally-tuned processing units* [3]. This type of network proposed in the paper was intended by Moody and Darken as an alternative network architecture to the popular Multi-Layer Perceptron (MLP). The nodes called *locally-tuned processing units* were designed to emulate the function of some biological neurons in human brain. So the architecture of RBFN was derived from two different approaches. Broomhead and Lowe deduced the network architecture from the RBF methods used mainly for interpolation problem, while Moody and Darken derived the network from the inspiration of human brain.

2.2 Definition

RBFN has a similar form to the MLP in that it is multi-layer, feed-forward network. However, unlike the MLP, the hidden units in the RBFN are different from the units in the input and output layers. The hidden units of a RBFN contain "Radial Basis Functions", a statistical transformation based on a Gaussian distribution from which the networks name is derived. Each basis function in the hidden layer has two parameters "centre" and "radius".

2.3 Radial Functions

Radial functions show the characteristic feature that their response decreases or increases with distance from a central point. A typical radial function is the Gaussian (Fig. 1). Its parameters are its centre c and its radius σ. A Gaussian radial function decreases monotonically with distance from the centre.

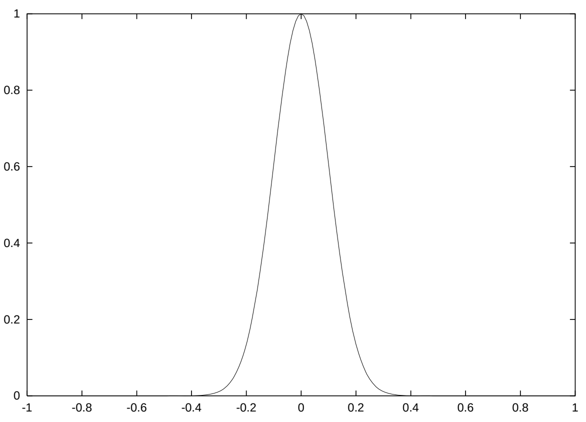


Figure 1: Gaussian $h = exp(-(c_i - x)^2/2\sigma_i^2)$.

2.4 Learning in RBFN

In order to obtain a solution for a given problem with a RBFN, it is essential to define the proper network architecture. The architecture of RBFN is determined by the number of nodes in each layer and the location of the function centres. The determination of the number of nodes in the input layer and output layer is easy but the determination of number of hidden nodes and location of the function centres require various techniques for different problems. The selection of the number of basis functions in the hidden nodes will have a significant influence on the performance of RBFN especially, when using large training data set. There are some techniques for determining the number of basis functions: a) One basis function for each training data point, b) Random selection of basis functions, c) Uniform selection of basis functions, d) The k-mean cluster algorithm, e) The Max-Min distance algorithm.

The most natural choice to find the number of basis functions is to let it equal to the number of training data points. The advantages of this solution are: it is simple; the choice of Gaussian centres is uniquely determined; it fully utilises each training simples. However, there are also many problems associated to this method, e.g. computation costs and over-fitting problem. On the other hand RBFN used Gaussian basis functions optimised with the k-mean clustering are not particularly well suited to neuro fuzzy modelling. We used uniform selected basis functions with constant radius and ability to change the position of Gaussian centres. This kind of RBFN is well suited to compare it with equidistant distributed B-FC MFs.

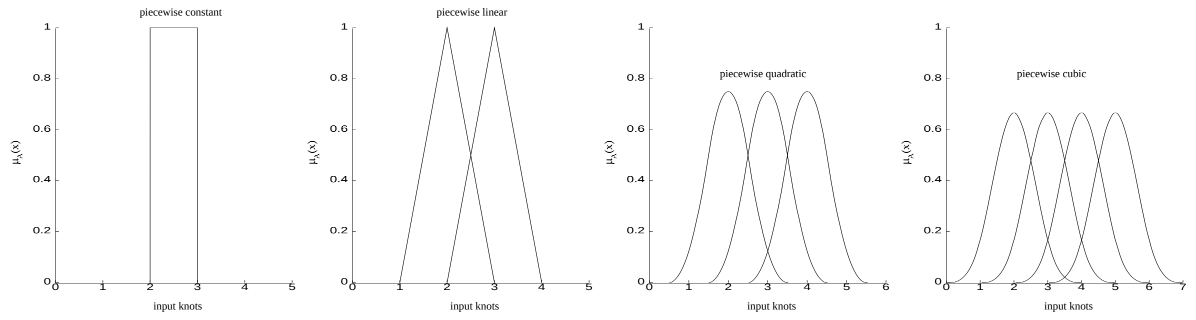
3 B-Spline Fuzzy Controller

3.1 Definition of B-Splines

B-Spline basis function are piecewise polynomial, producing models with a response of a desired smoothness. The universe of discourse of each input is divided into a number of subintervals, where each subinterval is delimited by so called *knot* which determines the position of each B-Spline. The order of these local polynomial is defined by the order of B-spline, denoted by k.

The B-spline $N_{i,k+1}$ of degree k with knots $\lambda_1, \ldots, \lambda_{i+k+1}$ is defined as (Fig. 2):

$$N_{i,k+1}(x) = (\lambda_{i+k+1} - \lambda_i) \sum_{j=0}^{k+1} \frac{(\lambda_{i+j} - x)_+^k}{\prod_{\binom{l=0}{l \neq j}}^{k+1} (\lambda_{i+j} - \lambda i + l)}. \tag{1}$$



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$

Figure 2: B-splines of order one to four.

3.2 Properties of B-Splines

B-spline basis functions possess many desirable properties ideal for representing fuzzy MFs. The important properties of the B-spline functions are :

Partition of unity:	$\sum_{i=0}^{n} N_{i,k}(x) = 1.$
Positivity:	$N_{i,k+1} \geq 0$ for all x
Local Support:	$N_{i,k+1} = 0$ if $x \notin [\lambda_i, \lambda_{i+k+1}]$
Recursion:	$N_{i,l+1}(x) = \frac{x-\lambda_i}{\lambda_{i+l}-\lambda_i} N_{i,l}(x) + \frac{\lambda_{i+l+1}-x}{\lambda_{i+l+1}-\lambda_{i+1}} N_{i+1,l}(x)$

3.3 Output of B-FC

The output of a single-input-single-output (SISO) B-FC is the unique representation of B-splines.

$$y = \sum_{i=0}^{n} d_i, N_{i,k}(x) \tag{2}$$

where $N_{i,k}(x)$ denote the linguistic terms of each input defined by B-splines and d_i are called control or de-Boor points. The variable n denotes the number of basis functions. This model can be interpreted as a fuzzy system of Tagaki-Sugeno type.

3.4 Learning in B-FC

The learning procedure of B-FC control points is based on the gradient descent approach. An error function is defined:

$$E = \frac{1}{2}(Desired\_Value - FC\_Output)^2 \qquad (3)$$

In each learning step the B-FC is modified with:

$$\delta_{y_i} = (Desired\_Value - FC\_Output) \cdot N_{i,k} \qquad (4)$$

3.5 Finding optimal Knot Vectors

The choice of knot vector has a significant influence on the B-spline function and hence on the resulting of B-FC. Fundamentally, three types of knot vectors are used: uniform, open uniform and non-uniform. In a uniform knot vector, individual knot value are evenly spaced. The open uniform knot vector has additional knot values at both ends which depends on the order k of B-spline. The task of finding optimal knot vectors to fit the training data becomes a non-linear minimisation problem. To solve this problem we follow a strategy of problem splitting. We first consider the underlying model $\delta(\lambda)$ and then compute the B-spline coefficients. To estimate the knot positions we use GA instead of using constrained least-square methods. GAs are both, theoretically and empirically proven to provide the means for efficient search, even in complex spaces. Therefore each individual, in example each B-FC with its special knot point distribution, represents one point in search space.

3.5.1 Knot Placement with the Genetic Algorithm

We used the GA introduced by Holland [2] but with the following modifications (see [6]):

- We used gray coding instead of standard binary code.
- Instead of using of fitness-proportional selection it has advantages to use tournament selection. This selection scheme draws ξ individuals $(2 \leq \xi \leq \mu)$ with a probability $\frac{1}{\mu}$ from the current population and copies the individual with the best fitness into the mating pool.
- To dodge the effect of the increasing probability along the descendent chromosome-string we used *uniform crossover*. This kind of crossover has no positional and a high distributional bias, so that a high blending rate between participant chromosomes is granted. This leeds to an algorithm producing permanently solutions which explore new locations by bridging even great distances of the search space.

To minimise $\delta(\lambda)$ each individual consists of n knot vectors, where n is the problem dimension. Each encoded knot vector consists of 32 knot points and a so called activation string of 32 bit length. Which knot points are in use to define the current model is encoded through the activation string. Activated knots are represented by 1 and inactivated knots are represented by 0. Every knot point is encoded by 16 bit and therefor each knot point can be placed on its concerning input interval *[a,b]* with an accuracy of $\frac{1}{2^{16}} \times (b - a)$. The fitness values for each individual is simply computed by determine the coefficients by solving the overdetermined linear system.

4 Architecture of RBFN and B-FC

The architecture of B-FC is determined by the number of the input/output variables and number of the membership functions which are used in the context of specifying linguistic terms. The number of membership functions can depend on the order of the B-splines. It is assumed that linguistic terms are to be used to cover the universe of an input variable. They are referred as real linguistic terms. In order to maintain the partition of unity, some more basis functions should be added at the both ends. They are called *marginal basis functions*, defining the *virtual linguistic terms*. In case of order 2, no marginal basis function is needed, where in case of order 3, two marginal basis functions are needed, one for the left end and another for the right end. In general case of order n ($n \geq 2$), $2n-4$ marginal basis functions are needed.

The architecture of the RBFN is determined by the number of the nodes in each layers and the location of the basis functions, where the input, hidden and output layer are fully connected. The hidden nodes activation values are determined by feeding the weighted sum of an input to a Gaussian function. Each hidden unit is associated with Gaussian function with centre and radius parameters. A comparison between Gaussian basis functions and B-splines is now given:

- Gaussian functions do not produce a partition of unity, while B-splines do. Hence the normalisation of Gaussian basis functions is liable to change their shape producing unexpected effects.
- Gaussian functions are infinitely differentiable and integrable wheres B-splines are piecewise polynomial.
- Gaussian functions do possess a localised response, but are not spatially compact, while B-splines are and hence a fixed number of basis functions contribute to an output.

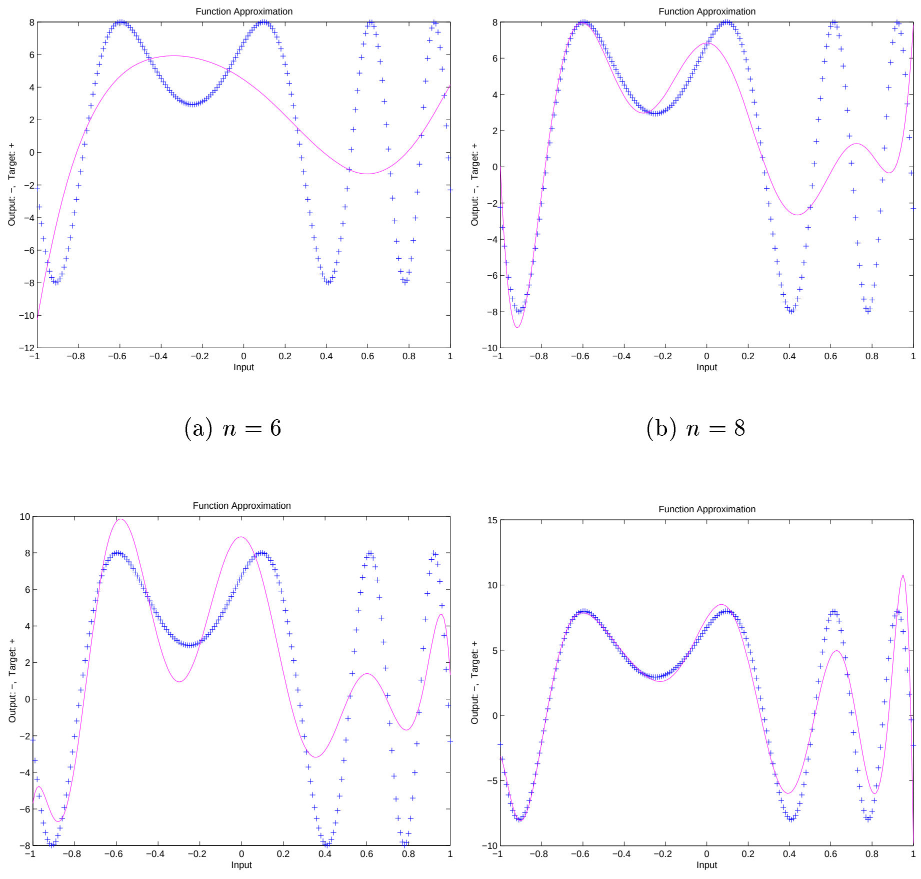
5 Simulation Results

We trained both RBFN and B-FC with different number of analytical functions to approximate different functions (see [4]):

$$\begin{aligned}
f_1(x) &= 3x(x-1)(x-1.9)(x-0.7)(x+18) && (5)\\
&\qquad \text{for} \quad -2 \leq x \leq 2 \\
f_2(x) &= 10\tan^{-1}\left(\frac{(x-0.2)(x-0.7)(x+0.8)}{(x+1.4)}\right) && (6)\\
&\qquad \text{for} \quad -1 \leq x \leq 1 \\
f_3(x) &= \frac{100(x+0.95)(x+0.6)(x+0.4)(x-0.1)(x-0.4)(x-0.8)(x-0.9)}{(x+1.7)(x-2)^2} && (7)\\
&\qquad \text{for} \quad -1 \leq x \leq 1 \\
f_4(x) &= 8\sin(10x^2+5x+1) && (8)\\
&\qquad \text{for} \quad -1 \leq x \leq 1 \\
f_5(x) &= 10\tan^{-1}\left(\frac{(x-0.2)(x-0.7)(x+0.8)}{(x+1.4)(x-1.1)x+0.7}\right) && (9)\\
&\qquad \text{for} \quad -1 \leq x \leq 1 \\
f_6(x) &= 10\left(e^{-5|x|}+e^{-3|x-0.8|/10}+e^{-10|x+0.6|}\right) && (10)\\
&\qquad \text{for} \quad -1 \leq x \leq 1 \\
& && (11)
\end{aligned}$$

The number of basis functions has a significant influence on convergence of both RBFN and B-FC. To show this influence we trained them with 6, 8, 10, 12, 14, 16, 18 and 20 basis functions to approximate the function f_4 and f_6.

Fig. 3 and 4 show the results of function approximation with RBFN in relation on number of Gaussian functions. Fig. 3 demonstrates the approximation to f_4 with 6, 8, 10 and 20 Gaussian



(a) $n = 6$ (b) $n = 8$

(c) $n = 10$ (d) $n = 20$

Figure 3: Demonstration of the effects of the number of basis function (n) on the convergence of RBFN for f_4. A sample of 200 data points are used.

functions and Fig. 4 the results of RBFN for f_6 with 6, 8, 10 and 20 Gaussian in comparing to results of B-FC with 10 and 20 B-splines.

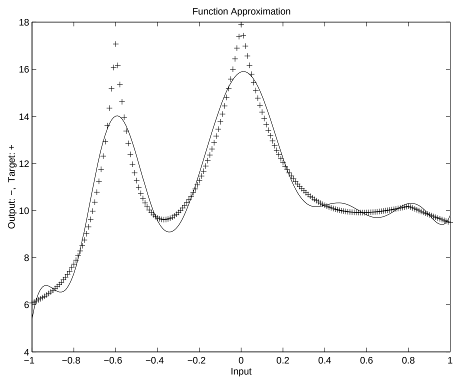
Table 1 represents the mean square error (MSE) of RBFN and B-FC with different number of basis functions. We used equidistant distributed B-splines of order 3 for B-FC. The radius of Gaussian functions was fixed with $\sigma = 1$. Typically the radius is chosen according to the equation:

$$\sigma = \frac{d}{\sqrt{2m}} \tag{12}$$

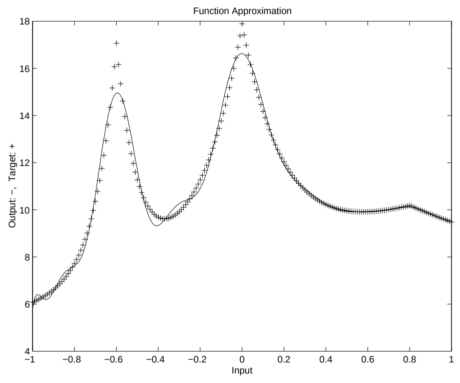
where m denotes the number of Gaussian basis functions and d is the maximal distance between the chosen centres. This $\sigma = 1$ should guarantee that the Gaussian functions are neither too flat nor too steep.

In Table 2 we represent the approximation results of B-FC with 12 equidistant distributed B-spline of order 3 and RBFN with 12 Gaussian basis functions.

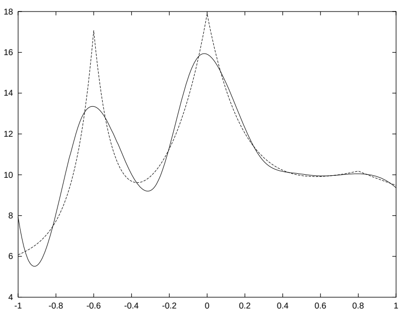
The convergence of B-FC can be improved by using the above described genetic algorithm to optimise the knot vectors, where parameter sittings were chosen as crossover probability = 0.75, mutation probability = 0.0005, $\mu = 40$ and $\xi = 3$. A maximum generation index of 200 was used as stop criterion. The MSE of each adapted B-FC represents the average MSE of 3 runs. To show these improvement we compared the B-FC with optimal knot vectors with RBFN (Table



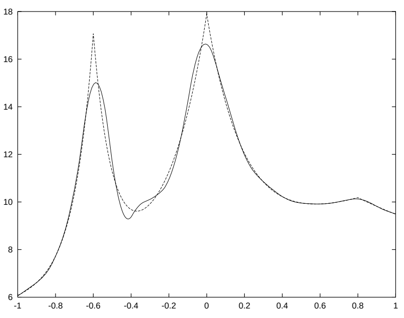
(a) RBFN with 10 Gaussian



(b) RBFN with 20 Gaussian



(c) B-FC with 10 B-splines



(d) B-FC with 20 B-splines

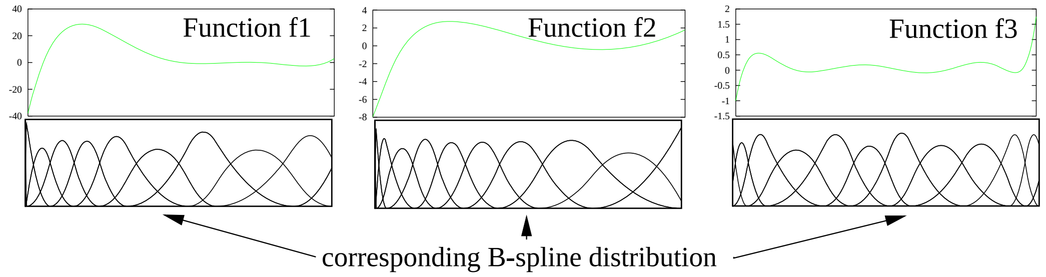
Figure 4: Demonstration of the effects of the number of basis function on the convergence of RBFN for f_6. (e) and (f) show the results of B-FC with 10 and 20 B-splines. A sample of 200 data points are used.

3) and the best results achieved in [4] (Table 4).

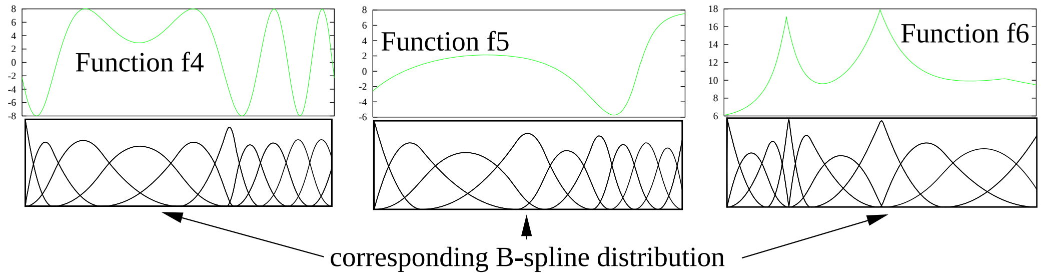
6 Conclusions

We have described the architecture of neuro-fuzzy models with Gaussian MFs, which form the popular RBFN and those with B-spline MFs, the B-FC. B-FC with equidistant distributed B-splines and B-FC with genetic adaption to optimal distribution of B-spline MFs are applied. Both neuro-fuzzy models were compared with RBFN theoretically and empirically.

By using B-splines as MFs in B-FC the design parameters are given by the order, number and distribution of basis functions. Due to the local support of B-splines it is possible to modify the data locally. It has been shown that genetic algorithm is capable to find optimal knot vectors and hence, less knot points are necessary to perform aa sccurate output than B-FC with equidistant distributed B-splines. This results in better generalisation abilities and smaller rule bases. We trained the RBFN and B-FC with different number of basis functions to approximate different functions. We scored each test in terms of MSE. The B-FC with equidistant distributed B-splines performed very well in most cases. In comparison with the results in [4], the B-FC with adapted B-splines provides the best results in all cases.



(a) f_1 - f_3



(b) f_4 - f_6

Figure 5: Demonstration of genetic adaption to optimal B-spline

References

[1] D.S. Broomhead and D. Lowe. *Multivariate functional interpolation and adaptive networks*, Complex Systems, 2:321-355, 1988

[2] Holland, J.H. Adaption in Neural and Artificial Systems. *Ann Arbor, MI: University of Michigan Press*,1975

[3] E. Moody and C. Darken, *Fast learning in networks of locally-tuned processing units*, Neural Computation 1, pp. 281-294, 1989.

[4] S. Mitaim and B. Kosko, *What is the Best Shape for a Fuzzy Set in Function Approximation*, Signal and Image Processing Institute, Department of Electricle Engineering-System, University of California, Los Angeles, California. Published in IEEE International Conference on Fuzzy Systems, 1996.

[5] J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems*, 13(2/3):257–285, Feb./Mar. 1998.

[6] J. Zhang, I. Renners and A. Knoll, *Genetic Adaptation to Optimal Membership Functions for Modelling with B-Splines*, Report AG Technical Computer Science, Faculty of Technology, University of Bielefeld, 1998.

n	6	8	10	12	14	16	18	20
RBFN	16.5	11	8.5	8	7.6	7	7.6	2.6
B-FC	17.3	15.34	10.98	4.95	1.4	2.346	0.76	0.17

Table 1: MSE of RBFN with different number of Gaussian and B-FC with equidistant distributed B-splines for f_4

Function	f_1	f_2	f_3	f_4	f_5	f_6
B-FC	0.02	0.0005	0.0008	4.9	0.04	0.6
RBFN	0.05	0.001	0.001	8	0.21	0.68

Table 2: MSE of B-FC with 12 equidistant distributed B-spline of order 3 compared with RBFN with 12 Gaussian

Function	f_1	f_2	f_3	f_4	f_5	f_6
B-FC	0.007	0.00003	0.000048	0.03	0.0002	0.012
RBFN	0.05	0.001	0.001	8	0.21	0.68

Table 3: MSE of B-FC with 12 adapted B-spline of order 3 compared with RBFN with 12 Gaussian

Function	Rules	Used Membership Function			
		Equidistant B-splines	Adapted B-splines	Best of [MiKo96]	RBFN
f_1	12	0.02	0.007	0.08	0.05
f_2	12	0.0005	0.00003	0.02	0.001
f_3	12	0.0008	0.000048	0.002	0.001
f_4	12	4.9	0.03	0.1	8
f_5	12	0.04	0.0002	0.01	0.21
f_6	12	0.6	0.012	0.1	0.68

Table 4: MSE results from [MiKo96] in comparison to results of an equidistant distributed B-spline network, results of a genetic modified B-spline network and RBFN.