International Journal of Information Sciences, 142, 2002, S. 227–248

EXTRACTING COMPACT FUZZY RULES BASED ON ADAPTIVE DATA APPROXIMATION USING B-SPLINES

J. ZHANG, S. KÖPER and A. KNOLL

Faculty of Technology, University of Bielefeld
33501 Bielefeld, Germany
Tel.: ++49-521-106-2951
Fax: ++49-521-106-6440
E-mail: zhang|skoeper|knoll@techfak.uni-bielefeld.de
http://www.techfak.uni-bielefeld.de/techfak/ags/ti/

Abstract

We discuss the importance of making a fuzzy controller human interpretable and give an overview of the existing models and structures for that purpose. We then summarise our approach to designing fuzzy controllers based on the B-spline model by learning. By using an optimal partition algorithm and using linguistic modificators like "between", "at most", "at least" etc., the rule base can be reduced to the minimum. This helps to avoid the over-fitting problem and improve the interpretability of the model. We tested the controller on different benchmark problems and achieved a rule compression ratio of up to 71%.

Key words: Rule extraction, Neuro-fuzzy system, Genetic Algorithms (GAs), interpretability

1 Introduction

Until recently, a large part of current science and technology was based on analytical methods which are usually specialised for pre-defined domains. However, for a human-being it is time-consuming to get acquainted with the model, to devise the model, even difficult to explain a model clearly to someone else. On this problem, physicist Richard P. Feynman mentioned "the way we have to describe nature is generally incomprehensible to us". Nevertheless Albert Einstein believed "it should be possible to explain the laws of physics to a barmaid".

Among the mechanisms to interprete the nature of a process like equations, tables, flow charts, stories, etc., fuzzy linguistic rules and relation descriptions are easy to understand. For building models with training data, certain types of fuzzy rule systems like the B-spline model [13] have been developed, which can approximate any low-dimensional input-output functions.

There are good reasons for making such a controller model symbolically interpretable:

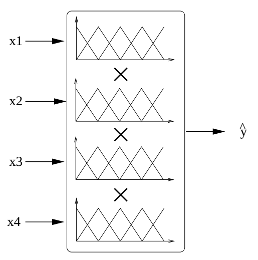
- Linguistic modelling provides a way of transferring skills from one human expert to other non-experts or to robots. The transfered rules and human knowledge can be used to accelerate the model-building and to patch the model in the case of data deficiencies.
- Automatic learning of transparent models makes the analysis, validation and supervision in the model/controller development easier. This way, a large part of design expenses can be shifted from humans to the computer.
- Transparent models will have wide applications in decision-support systems. In the next years, most of the control of complex systems will still be semi-autonomous. "Human-in-the-Loop" is based on compact and summarising descriptions of a system model.

New control and modelling approaches will make good use of the rapid increase of computation power and memory brought about by recent computer technology. Advances have been made in the theory and applications of neural networks and fuzzy rule based systems to the control of physical systems which cannot be adequately modelled by (linear) differential equations. The fuzzy rule description of a system has the advantage over neural networks that it is not just a "black-box". Neuro-fuzzy models integrate automatic feature extraction and learning of membership functions into a fuzzy control structure. Together with optimisation algorithms, which deal with local minima, semi-automatic procedures can be designed for constructing non-linear models and controllers which can be understood by human users.

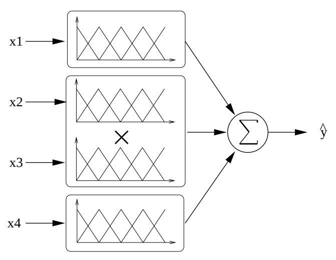
2 State of the Art

2.1 Structure of Interpretable Models

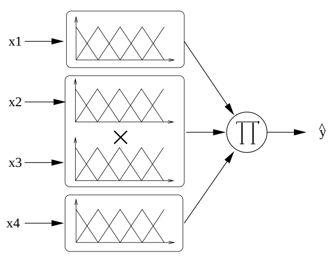
Variants of fuzzy rule systems are: additive, multiplicative and hierarchical. Fig. 1 and Fig. 2 illustrate these structures. The solution with hierarchical structure assumes that the input information can be classified into groups [12]. Within each group the inputs determine an intermediate variable, they can be decoupled from inputs of other groups. To realise such a grouping, usually heuristics based on the fusion of physical sensors have to be applied.



(a) Complete fuzzy system (5^4 rules).

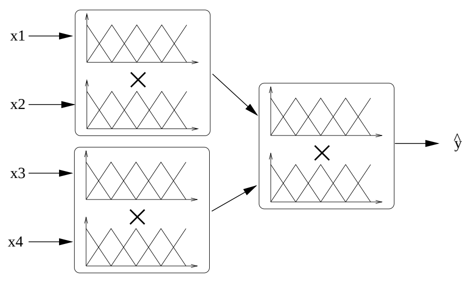


(b) Additive fuzzy system (35 rules).

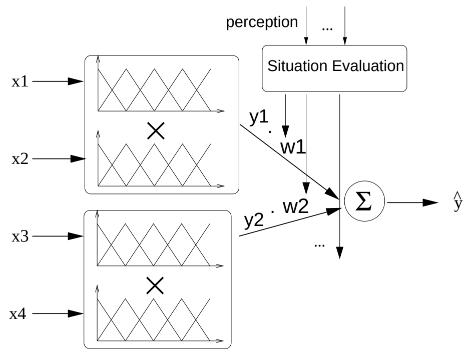


(c) Multiplicative fuzzy system (35 rules).

Fig. 1. Fuzzy system with 4 inputs and each with 5 linguistic terms.



(a) Hierarchical fuzzy system with four inputs, each with 5 linguistic terms – resulting in 75 rules.



(b) Behaviour blending using a two-step hierarchy. "Situation Evaluation" uses rules to determine the weight of each monolithic controller [14].

Fig. 2. Hierarchical fuzzy systems.

As application areas grow, the *systematic* design of an optimal fuzzy controller becomes more and more important. Classical fuzzy controllers of the Mamdani type [7] are based on the idea of directly using symbolic rules for diverse control tasks. Another important type of fuzzy controllers is based on the TSK (Takagi-Sugeno-Kang) model [10]. Recently, TSK type fuzzy controllers have been used for function approximation and supervised learning [11]. However, it is pointed out that the TSK model is a black-box based on a multi-local-model. In the following section, we describe an approach that can build membership functions (MFs) for linguistic terms of the IF-part systematically, then optimize them by genetic algorithms. Our approach is based on the B-spline model which can be classified as a zero-order TSK model. However, we define linguistic terms for input variables with B-spline basis functions and for output variables with singletons. Such a method requires fewer parameters than other set functions such as trapezoid, Gaussian function, etc. The output computation is very simple and the interpolation process is transparent. We also achieved good approximation capabilities and rapid convergence of the B-spline controllers.

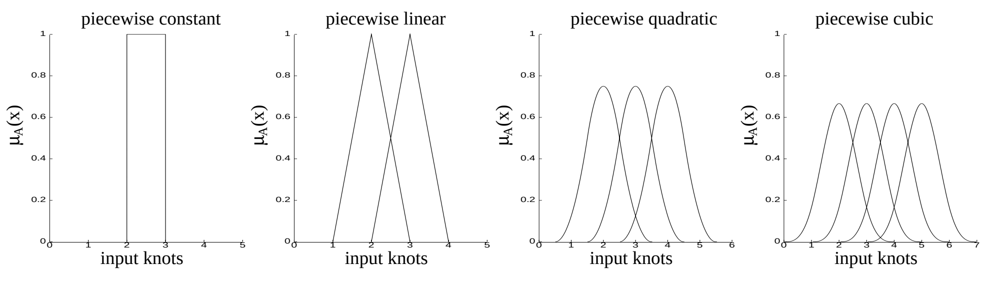
3 Constructing Fuzzy Controllers with B-Splines

3.1 Basis Concept

Our B-spline model provides an ideal implementation of CMAC ("Cerebellar Model Articulation Controller") proposed by Albus [1]. The CMAC model provides an neuro-physiological interpretation of the the B-Spline model. B-spline models employ piecewise polynomials as MFs. The universe of discourse of each input is divided into a number of subintervals, where each subinterval is delimited by breakpoints called *knots* which determine the appearance and position of each B-spline.

3.1.1 Definition of Univariate B-Splines

The B-splines are employed to specify the linguistic terms, and knots are chosen to be different from each other (periodical model). Visually, the selection of k (the order of the B-splines) determines the following factors of the fuzzy sets for modeling the linguistic terms.



(a) Order 1 (b) Order 2 (c) Order 3 (d) Order 4

Fig. 3. Univariate B-splines of order 1–4.

Assume x is a general input variable of a control system that is defined on the universe of discourse $[x_1, x_m]$. Given a sequence of ordered parameters (knots): $x_1, x_2, \ldots,$, the ith B-spline $N_{i,k}$ of order k (degree $k-1$) is recursively defined as follows (Fig. 3):

$$N_{i,k}(x) = \begin{cases} \begin{cases} 1 & \text{for } x \in [x_i, x_{i+1}) \\ 0 & \text{otherwise} \end{cases} & \text{if } k = 1 \\ \frac{x-x_i}{x_{i+k-1}-x_i} N_{i,k-1}(x) + \frac{x_{i+k}-x}{x_{i+k}-x_{i+1}} N_{i+1,k-1}(x) & \text{if } k > 1 \end{cases} \tag{1}$$

with $i = 1, \ldots, m-k$. Therefore m knots $x_i (i =, \ldots, m)$ form $l = m - k$ B-splines (Fig. 4).

3.1.2 Properties

Recursive definition is one basic feature of B-splines, which enables the generation of basis functions of arbitrary orders with the incremental smoothness for a given set of knots. The other most important properties of B-splines, with respect to modeling and control are:

Partition of unity:	$\sum_{i=0}^{l} N_{i,k}(x) = 1$.
Positivity:	$N_{i,k}(x) \geq 0$ for all x.
Local support:	$N_{i,k}(x) = 0$ for $x \notin [x_i, x_{i+k}]$.
C^{k-2} *continuity:*	If the knots $\{x_i\}$ are pairwise different from each other, then $N_{i,k}(x) \in C^{k-2}$, i.e., $N_{i,k}(x)$ is $(k-2)$ times continuously differentiable.

Fig. 4 shows the non-uniform B-splines which are computed with unevenly distributed knots. To compactly model and control a system, it is desirable that the minimal number of basis functions are used. Non-uniform B-splines are the important model in these applications. They coincides with the psychological investigations that human linguistic terms are often non-symmetrical.

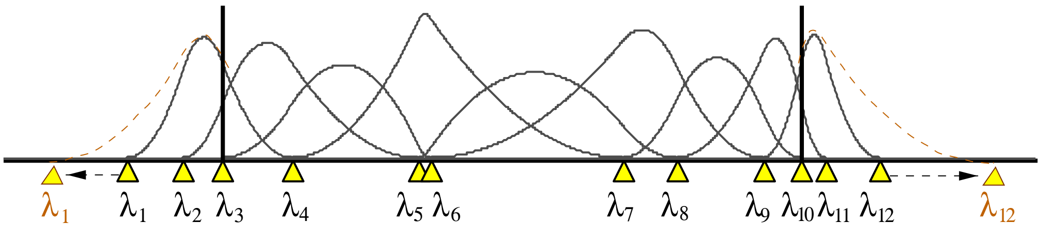


Fig. 4. Nine B-splines of order 3 defined over 12 non-uniformly distributed knots.

3.2 Lattice

In our previous work, we compared splines and a fuzzy controller with *single-input–single-output* (SISO) structures. In the following, the MISO controller is considered.

Fig. 5 illustrates the partition of a two-dimensional B-spline model with 8 MFs on each uniformly subdivided input interval and the activated B-splines (slightly shaded) for a given input. Since learning one new part of the input space affects only a given number of controller response values (darkly shaded area of Fig. 5), fast on-line learning can be implemented. Due to these advantages, B-spline models are proposed to be applied in control systems and will be denoted as B-spline Fuzzy Controllers [13]. By using the B-spline model the approximation ability is only limited by the number of knots distributed over the input intervals. Regarding that most observed data are disturbed to a certain degree, the problem of over-fitting may occur. GA-optimized B-spline models are a promising approach to find sparse models, which are able to bridge the gap between high bias and high variance of a model.

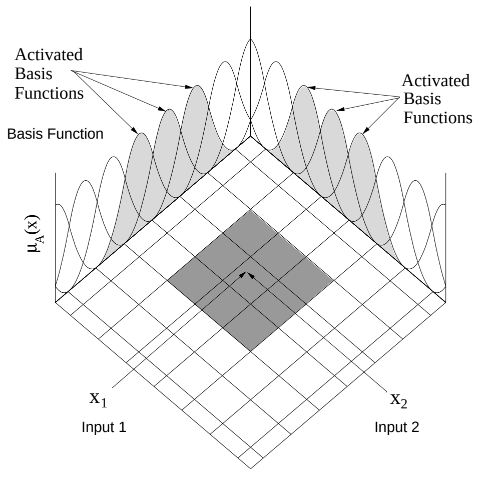


Fig. 5. The B-spline model – a two-dimensional illustration.

Each n-dimensional rectangle $(n > 1)$ of the lattice is covered by the j^{th} multivariate B-spline $N_k^j(x)$ which is formed by taking the tensor product of n univariate B-splines:

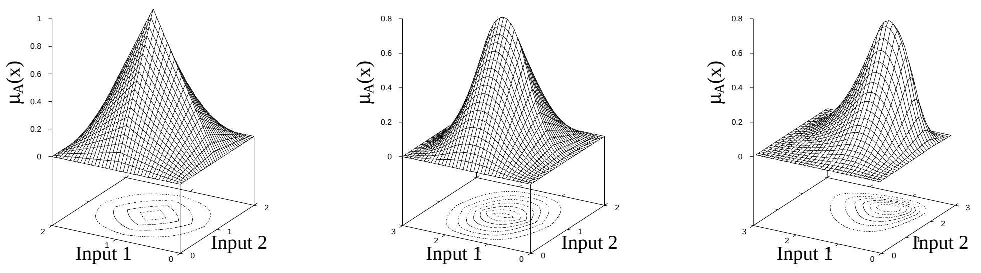
$$N_k^j(x) = \prod_{j=1}^{n} N_{i_j,k_j}^j(x_j) \tag{2}$$

Therefore the shape of each B-spline, and thus the shape of multivariate ones (Fig. 6), is implicitly set by their order and their given knot distribution on each input interval.

3.3 Use B-Splines as Fuzzy Controller

Under the following conditions, the computation of the output of such a fuzzy controller is equivalent to that of a *general B-spline hypersurface* [13].

(1) periodical B-spline basis functions are used as MFs for inputs,



(a) Tensor product of two, order 2 univariate B-splines.

(b) Tensor product of one order 3 and one order 2 univariate B-splines.

(c) Tensor product of two univariate B-splines of order 3.

Fig. 6. Bivariate B-splines formed by taking the tensor product of two univariate B-splines.

(2) fuzzy-singletons (called *control vertices*) are used as MFs for outputs,
(3) "product" is used as fuzzy-conjunctions,
(4) "centroid" is used as defuzzification method,
(5) "virtual linguistic terms" are added at both ends of each input variables, and
(6) the rule base for the "virtual linguistic terms" is extended by copying the output values of the "nearest" neighborhood.

The computation of the output of such a fuzzy controller is equivalent to that of a general B-spline *hyper-surface*. Generally, if we consider a MISO system with n inputs $x_1, x_2, \ldots, x_n$, a $rule(i_1, i_2, \ldots, i_n)$ with the n conjunctive terms in the premise is given in the following form:

$$\begin{array}{ll} \text{IF} & (x_1 \text{ is } N^1_{i_1,k_1}) \text{ and } (x_2 \text{ is } N^2_{i_2,k_2}) \text{ and } \ldots \text{ and } (x_n \text{ is } N^n_{i_n,k_n}) \\ \text{THEN} & y \text{ is } c_{i_1 i_2 \ldots i_n} \end{array}$$

where:

- x_j: the j^{th} input $(j = 1, \ldots, n)$,
- k_j: the order of the B-splines used for x_j,
- $N^j_{i_j,k_j}$: the i^{th} linguistic term of x_j defined by univariate B-splines,
- $i_j = 1, \ldots, l_j$: represents the index of the linguistic term of x_j,
- l denotes the number of B-splines and
- $c_{i_1,i_2,\ldots,i_n}$: the coefficients of the rule $(i_1, i_2, \ldots, i_n)$, called *control vertices* in the following[1].

Then the output y of a MISO fuzzy controller is:

[1] Corresponding to *de Boor points* in CAGD.

$$y = \frac{\sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \left(c_{i_1,\ldots,i_n} \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j) \right)}{\sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j)} \tag{3}$$

$$= \sum_{i_1=1}^{l_1} \cdots \sum_{i_m=1}^{l_n} \left(c_{i_1,\ldots,i_n} \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j) \right) \tag{4}$$

This is called a *general NUBS hypersurface*, which possesses the following properties:

- If the B-splines of order $k_1, k_2, \ldots, k_n$ are employed to specify the linguistic terms of the input variables $x_1, x_2, \ldots, x_n$, it can be guaranteed that the output variable y is $(k_j - 2)$ times continuously differentiable with respect to the input variables $x_j, j = 1, \ldots, n$.
- If the input space is partitioned fine enough and at the correct positions, the interpolation with the B-spline hypersurface can reach a given precision.

The transformation from (4) to a rule base is conceptually important because it provides a construction method for data approximation using fuzzy controllers. The advantage of the fuzzy control idea over the pure B-spline interpolation lies mainly in its linguistic modeling ability: interpolation data can be prepared by using natural language with the help of expert knowledge. Furthermore, the interpolation procedure becomes transparent because it can also be interpreted in fuzzy logic “IF–THEN” form.

3.4 Generating the THEN-Part

Determining optimal coefficients of a B-spline model with fixed knot vector is generally simpler than finding the optimal knot vectors. Applying methods of B-spline theory in CAGD (Computer Aided Graphic Design), the coefficients can be estimated by solving an overdetermined linear system. Another method based on gradient descent can also be used. Although an iterative solution, this learning method is conceptually more easily understandable. Based on this learning method, unsupervised learning procedure of B-spline coefficients can also be derived.

In the gradient descent approach, fuzzy singletons represented by control points can be initialised with the values acquired from expert knowledge. These parameters will be fine-tuned by a learning algorithm.

For supervised learning, we show in the following that the squared errors with respect to control points are convex functions. Therefore, rapid convergence for supervised learning is guaranteed. The control space changes locally due to the “local support” property of B-functions while the control points are modified. Based on this feature, the control points can be optimised gradually, area-by-area.

Assume that $(x_1^r, \ldots, x_n^r, y^r_{desired})$ is a set of training data. The output value computed by a BFC is denoted with $y^r_{computed}$. By defining the Mean-Square Error (MSE) criterion as:

$$E = \frac{1}{2} \cdot (y_{computed} - y_{desired})^2 \equiv \text{MIN}, \tag{5}$$

the derivation of each coefficient $c_{i_1,\ldots,i_k}$ is:

$$\Delta c_{i_1,\ldots,i_k} = -\epsilon \frac{\partial E}{\partial c_{i_1,\ldots,i_k}} = \epsilon (y^r_{computed} - y^r_{desired}) \cdot \prod_{j=1}^{n} N_{i_j,n_j}(x^r_{i_j}), \tag{6}$$

where ϵ denotes the *learning rate*. Since the second partial derivation of $c_1^r, \ldots, c_k^r$ is always positive, the error function (5) is convex in weight space. If the inputs $(x_1^r, \ldots, x_n^r)$ are *linearly independent*, there exists, due to the convexity of the MSE performance surface, only one global minimum and no local minima. On the other hand, if the autocorrelation matrix is singular, which occurs when the inputs x_r are *linearly dependent*, there exists an infinite number of global minima (Fig. 7), but still no local minima, in weight space.

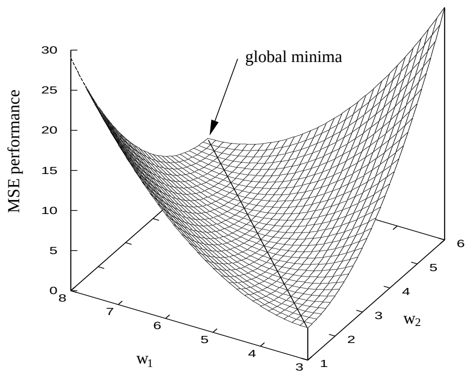


Fig. 7. A singular MSE performance surface.

3.5 Adaptive Modelling of the IF-Part

Based on the granularity of the input space and the distribution of extrema in the control space (if known), the fuzzy sets can be initialised using the recursive computation of B-splines. These fuzzy sets based on non-uniform B-splines can be further adapted during the generation of the whole system by using GAs.

By freeing the knots, the task of finding control points and accurate knot vectors to fit 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 $\delta(\lambda)$ of the controller and then compute the control points to minimise $\delta(c)$. Instead of using constrained least-square methods (constrained because of avoiding to "ride" on the gradient edge of coincident knots [5]), we try to estimate the knots by using GAs, because GAs are both theoretically and empirically proven to provide the means for efficient search, even in complex spaces [3]. Therefore each individual, in the example each B-spline controller with its special knot distribution, represents one point in search space.

3.5.1 The Genetic Algorithms

We applied the basic GA introduced by Holland [4] with some modifications as follows (Fig. 8):

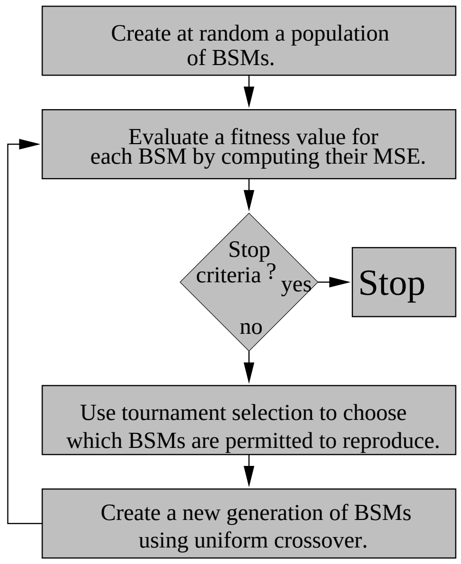

Create at random a population of BSMs.
Evaluate a fitness value for each BSM by computing their MSE.
Stop criteria ?
yes
Stop
no
Use tournament selection to choose which BSMs are permitted to reproduce.
Create a new generation of BSMs using uniform crossover.

Fig. 8. Flowchart of the applied genetic algorithm.

- We used *gray coding* instead of standard binary code while representing coded chromosomes, a common modification.
- To bypass the undesirable effect of the increasing probability along the descendent chromosome-string to receive a changed allele (bit) (thus the conjointly heredity of genes decreases when the distance of their position increases) while using n-point crossover, 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 leads to an algorithm producing permanently solutions which explore new locations by bridging even great distances of the search space.

- Instead of using fitness-proportional selection it is advantageous to use *tournament selection*. This selection schema 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. Besides saving computational power as a result of no need to sort the population (as in ranking based selection schemes), it is easier to bias the *takeover time*.

3.5.2 Chromosome Encoding for Knot Placement

To minimise $\delta(\lambda)$ each individual consists of n knot vectors, where n is the problem dimension. Each encoded knot vector consists of 32 knots and a so-called *activation string* of 32 bit length. Which knots 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.

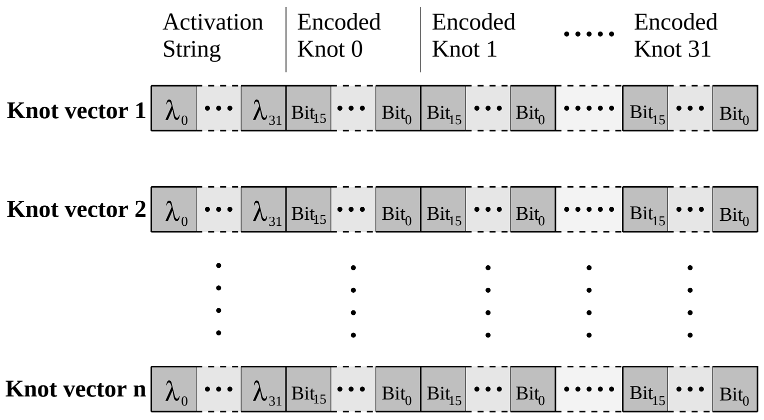


Fig. 9. Encoded B-spline model.

Every knot is encoded by 16 bit (see Fig. 9) and therefore each knot can be placed on its respective input interval $[a, b]$ with an accuracy of $\frac{1}{2^{16}} \times (b - a)$. The fitness values for each individual are simply computed by determining the control points of the controller. Using these control points the mean square error is evaluated and the fitness for one individual is set equal to $\frac{1}{MSE}$.

4 Optimal Number of Membership Functions

An easy way to reduce the number of fuzzy rules is to minimise the number of linguistic terms. However, when decreasing the number of membership functions the approximation error increases. We studied the problem how many linguistic terms are needed to minimise both the number of linguistic terms *and* the approximation error.

We analysed the approximation of twelve functions [8] with several fuzzy controllers. Six functions have one input, three have two inputs and three have three input dimensions. For approximating the functions, several fuzzy controllers with different numbers of linguistic terms were trained with the algorithms described in section 3. Then the approximation errors were analysed.

An example shows the approximation results of function g_2. This function is defined in (Eqn. 7).

$$g_2(x,y) = f_4(x)\cdot 2\left(\mathrm{e}^{-(\frac{y-0.1}{0.25})^2} - 0.8\mathrm{e}^{-(\frac{y+0.75}{0.15})^2} - 0.4\mathrm{e}^{-(\frac{y-0.8}{0.1})^2}\right), \quad \text{with} -1 \le x, y \le 1 \qquad (7)$$

Fig. 10 shows the relationship between the number of linguistic terms and the mean square error of the approximation. The approximation error function has a typical shape. While increasing the number of linguistic terms the error first decreases fast, later slowly. A good compromise between a minimum number of linguistic terms and a minimum approximation error is using eight terms for input x and three terms for input y. More terms would not improve the approximation substantially, less terms would increase the error sharply.

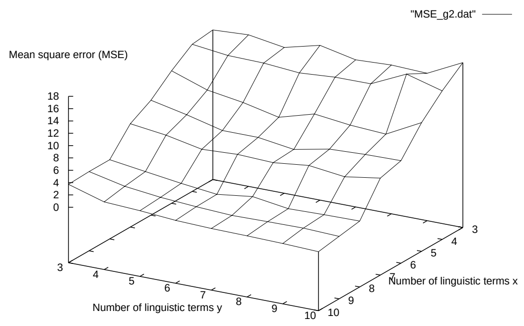


Fig. 10. Relationship between the number of linguistic terms and the mean square error (MSE) of the fuzzy controller (function $g_2(x,y)$)

.

Fig. 11 about function g_2 and there projections on the x- and y-axis in fig. 12 and 13 show why we get this numbers for the optimal number of linguistic terms. The extrema of the function lie on a grid. If the extrema are projected on the axes, several extrema will cover each other. On the x-axis we find eight groups of extrema and on the y-axis three groups that are just the same numbers we found for the optimal number of linguistic terms.

Furthermore it is interesting to observe how the genetic algorithms arranges the membership functions on the input axises (figure 14 and 15). The membership functions cover nearly all the groups of extrema we found in Fig. 12 and 13. They lie on the same grid as the function's extrema. Lastly we point out that in the same way a human would describe this function with fuzzy rules without having a fuzzy controller. For each extrema he would use one fuzzy rule. For the other functions we found exactly the same results.

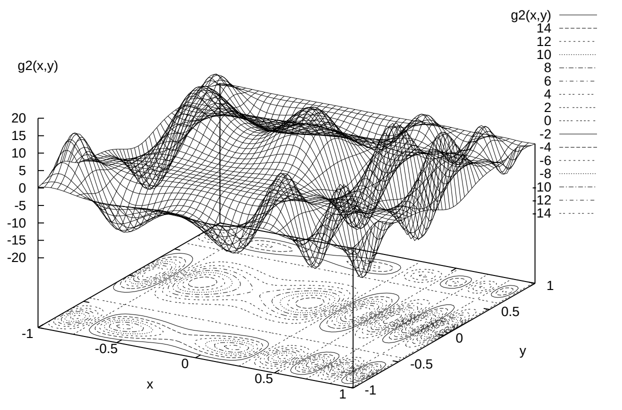
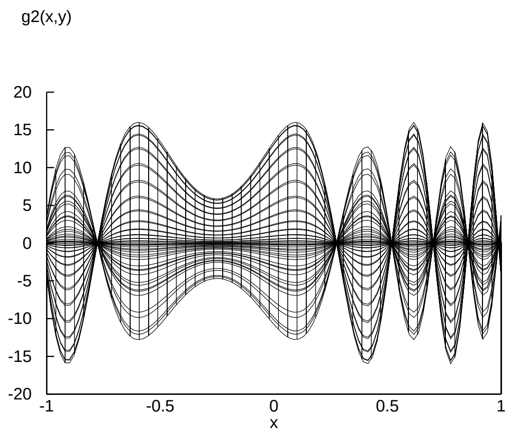


Fig. 11. The function g_2.



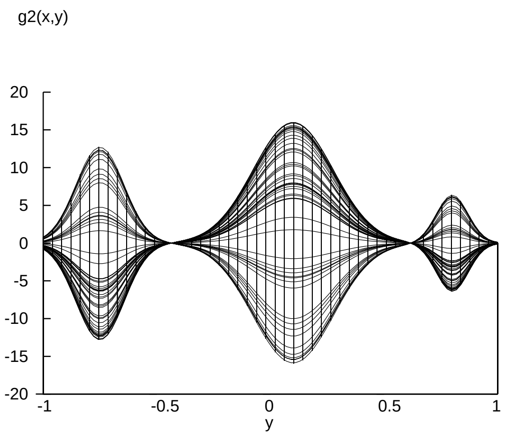
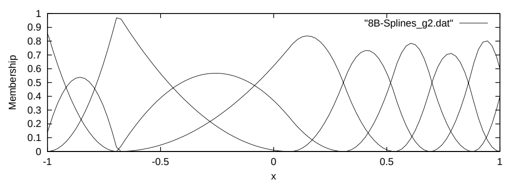


Fig. 12. Projection of g_2 to the x-axis. Fig. 13. Projection of g_2 to the y-axis.



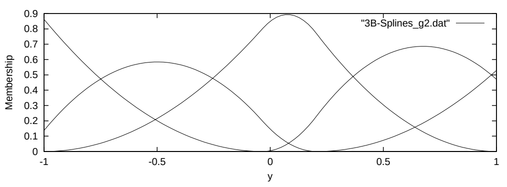


Fig. 14. Arrangement of eight B–spline membership functions on the x-axis. Fig. 15. Arrangement of three B–spline membership functions on the y-axis.

5 Optimal Partition Algorithm

If an application demands a high precision, a lot of fuzzy rules will be necessary. This large number of rules will have a negative effect on the interpretability of the model. This section describes an optimal partition algorithm, which helps to explain how a system with a large number of rules can be best interpreted.

5.1 Basic Idea

A fuzzy controller divides the universe of inputs and outputs in fuzzy sets. This fuzzy sets are arranged e. g. in "small", "middle" and "large". These information is useful to reduce the fuzzy rule base. Table 1 shows a simple example of a fuzzy controller with two inputs [2]. Each input is covered by five fuzzy sets ("NB", "NS", "Z", "PS", "PB").

$e\backslash\dot{e}$	NB	NS	Z	PS	PB
NB	NB	NB	NB	NS	Z
NS	NB	NB	NS	Z	PS
Z	NB	NS	Z	PS	PB
PS	NS	Z	PS	PB	PB
PB	Z	PS	PB	PB	PB

Table 1
Example controller.

A simple way to reduce this rule base is to combine neighbouring rules with same consequences, e. g. the three rules

IF e IS *'NB'* AND $\dot{e}$ IS *'NB'* THEN f IS *'NB'*
IF e IS *'NS'* AND $\dot{e}$ IS *'NB'* THEN f IS *'NB'*
IF e IS *'Z'* AND $\dot{e}$ IS *'NB'* THEN f IS *'NB'*

can be replaced by the rule

IF e IS *'at most Z'* AND $\dot{e}$ IS *'NB'* THEN f IS *'NB'*.

Another possibility is to replace the four rules

IF e IS *'NB'* AND $\dot{e}$ IS *'NB'* THEN f IS *'NB'*

IF e IS *'NS'* AND $\dot{e}$ IS *'NB'* THEN f IS *'NB'*

IF e IS *'NB'* AND $\dot{e}$ IS *'NS'* THEN f IS *'NB'*

IF e IS *'NS'* AND $\dot{e}$ IS *'NS'* THEN f IS *'NB'*

by the rule

IF e IS *'at most NS'* AND $\dot{e}$ IS *'at most NS'* THEN f IS *'NB'*.

In this example the modifier "at most" is used to combine several rules into one rule. Other modifiers are "at least", and "between". For a definition of these modifiers see [2].

There are several possibilities to combine neighbouring rules in a rule table. Not all of these possibilities lead to the minimum of rules. The question is how to find the optimal partition of the rule base. One step to answer this question are the following three properties of rules that can be combined to one rule:

(1) the same consequences,
(2) direct neighbourhood, and
(3) the rules have to form rectangular regions in the rule table.

The result of property 1 and 2 is that the search can be restricted to regions with same consequences. These regions form polygons in the rule table which are composed of squares (one square per rule). This polygons are called rectilinear polygons. If this polygon is a rectangle, than it will be a valid rule (feature 3). The way to reduce the rule base with two inputs to the minimum is to find a partition of all of these polygons in a minimum number of rectangles. If there are more then two inputs there will be hyper-polygons composed of hyper-cubes. For this hyper-polygons a partition in hyper-cuboids is searched.

5.2 Minimum partition of a polygon in rectangles

There are several algorithms to solve the problem of partitioning a polygon in a minimum number of rectangles [6,9]. The main idea is: The difference between a rectilinear polygon and a rectangle is that a complex polygon is not convex. Every convex rectilinear polygon is a rectangle. The goal is to intersect the polygon horizontally and vertically that all concave vertices are removed. Every intersection increases the number of rectangles by one and decreases the number of concave vertices by one if the cut goes through one concave vertice or by two if the cut goes through two concave vertices. The optimal partition will be reached if first a *maximum* of cuts through two concave vertices and then the cuts through one concave vertices are made.

The optimal partition of a rule base takes the following steps:

(1) Find all connected areas with the same consequences in the rule table. Then a set of polygons is generated.
(2) For all polygons: find the maximum number of cuts through two concave vertices.
(3) Perform these cuts.
(4) Intersect the polygons through the remaining concave vertices.
(5) For all rectangles: find the corresponding fuzzy rules.

Figure 16 shows all possible partitions of an example polygon.

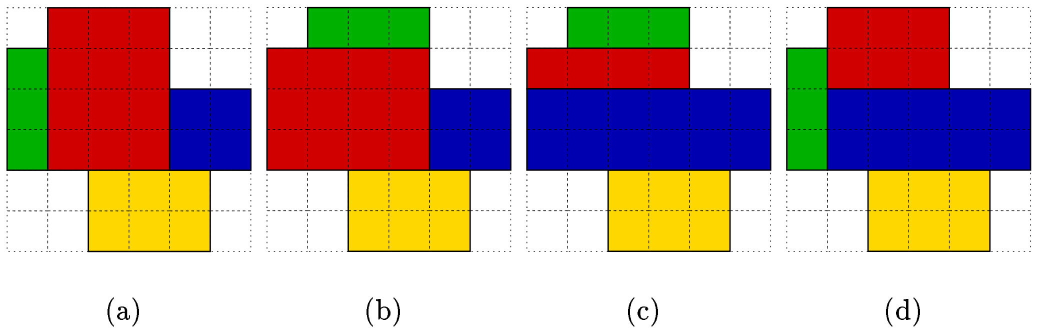


Fig. 16. Minimum partitions of a polygon.

5.3 Applying the Algorithm on B-Spline Fuzzy Systems

The described algorithm is easy to apply to fuzzy rule bases whose consequences are real fuzzy sets (like the rules of Mamdani controllers). The consequences of B-spline controller rules are fuzzy singletons. Every singleton is learned by a machine learning algorithm and the possibility that two singletons have the same value is very small. Before the algorithms can be applied to a B-spline rule base the number of possible fuzzy singletons must be reduced, e. g. with a clustering algorithm like the fuzzy c-means algorithm.

The second problem is that if B-spline systems with order three or higher are used, the singletons will not be the interpolation points of the curve (see Fig. 17). If the rule base shall be interpreted it will be reasonable to use the interpolation points. These points can easily be calculated.

Before the partition algorithm can be applied to a B-spline rule base the fuzzy singletons must be replaced by the interpolation points, then the interpolation points must be clustered.

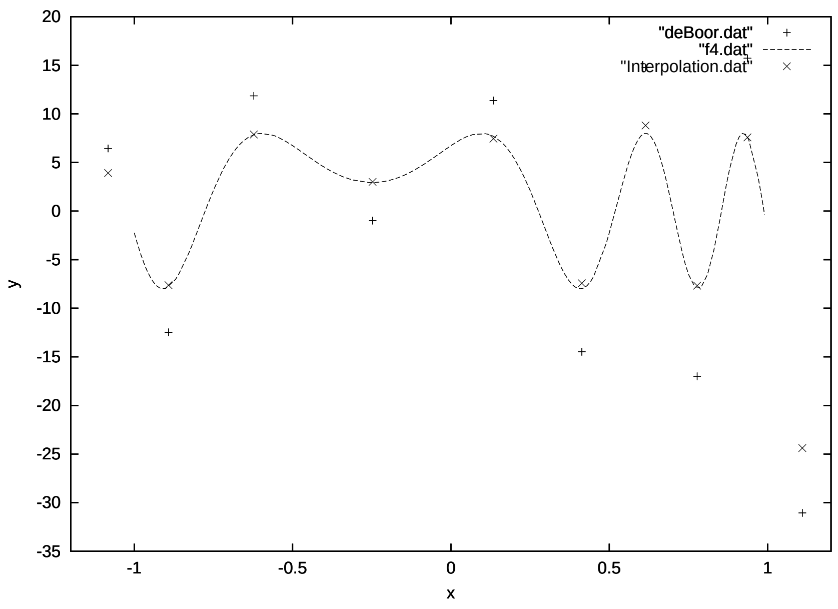


Fig. 17. Function $f_4(x)$ (dashed), de Boor-points (fuzzy singletons) (+), interpolation points (×).

Table 2 shows the original rule table, table 3 the rules with interpolation points as consequences and table 4 the resulting rule table with clustered consequences.

IF x IS	LM	VT	T	VS	S	L	VL	H	VH	RM
THEN y IS	6.4	-12.5	11.9	-1.0	11.4	-14.5	14.8	-17.0	15.7	-31.1

Table 2
Rule table for function $f_4(x)$

IF x IS	LM	VT	T	VS	S	L	VL	H	VH	RM
THEN y IS	3.9	-7.6	7.9	3.0	7.4	-7.4	8.8	-7.7	7.6	-24.4

Table 3
Rule table for function $f_4(x)$ consequences are interpolation points

IF x IS	LM	VT	T	VS	S	L	VL	H	VH	RM
THEN y IS	L	S	VL	L	VL	S	VL	S	VL	VS

Table 4
Rule table for function $f_4(x)$ consequences are clustered interpolation points labeled with linguistic identifiers

5.4 Examples

Two examples show how a fuzzy rule base can be simplified. The first example is the well-known MPG-Problem [2]. We trained a uniform B-spline system with two inputs (year, weight) and five fuzzy sets per input. The fuzzy singletons were replaced by the interpolation points and then clustered in five fuzzy sets. Table 5 shows the rule base with 25 rules.

This rule base can be reduced with our partition algorithm to 12 rules. This is a saving of 52%. Table 6 shows the reduced rule base.

year \ weight	VS	S	M	L	VL
VS	M	M	S	S	VS
S	VL	M	S	S	VS
M	VL	M	S	S	M
L	VL	L	M	S	S
VL	VL	L	L	S	VS

Table 5
Rule base for the MPG-problem.

year \ weight	VS	S	M	L	VL
VS	M	M	S	S	VS
S	VL	M	S	S	VS
M	VL	M	S	S	M
L	VL	L	M	S	S
VL	VL	L	L	S	VS

Table 6
Reduced rule base for the MPG-problem.

The second example is the inverse kinematics of a two-joint planar robot. For the angle of the joints a non-uniform B-spline system was trained. Each input was covered by 11 fuzzy sets. With the partition algorithm the rule base of 121 rules was reduced to 35 rules. This is a saving of 71%. Table 7 shows the reduced rule base.

[2] The data are avaliable at
`ftp://ics.uci.edu/pub/machine-learning-databases/auto-mpg`

y \ x	LM	VT	T	VS	S	M	L	VL	H	VH	RM
LM	M	M	S	S	S	S	S	S	S	S	M
VT	M	S	S	S	M	M	M	M	M	S	S
T	S	S	S	S	M	S	M	M	M	M	S
VS	S	S	S	S	M	M	M	M	M	M	M
S	S	S	S	S	S	VS	L	L	L	M	M
M	VL	VS	S	S	S	VL	L	L	L	M	M
L	VS	VL	S	VS	L	L	L	L	L	M	M
VL	M	L	L	L	L	L	L	L	L	M	M
H	L	L	L	L	L	L	L	L	L	M	M
VH	M	M	L	L	L	L	L	L	M	M	M
RM	M	M	M	M	M	L	L	M	M	M	M

Table 7
Grouped rule base for the inverse kinematics, joint 1.

6 Future Research Topics

Integration of neural networks, fuzzy systems, genetic algorithms, chaos theory with the classical probability theory and control methods will play a central role in the future research. We list some important ones:

- Optimal input selection: factor analysis/synthesis, tracking focus of interests. Extention of sensor capability by using up-to-date information technologies: software robots in WWW, linguistic modelling of human perception and sensor fusion so that information which is difficult to measure, incomplete or noisy can be perceived.
- Increasing the capability and quality of reinforcement signals and fitness evaluation of the learning system. Integrating symbolic sparse coding, granular computing, fuzzy set, rough set to enable the arbitrary transition between digital measurements and concepts.
- Learning from analytical models and generalise.

The model and approaches described in this paper paves the way for layered-learning to build models of a wide range of modeling and control problems.

References

[1] J. S. Albus. A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC). *Transactions of ASME, Journal of Dynamic Systems Measurement and Control*, 97:220–227, 1975.

[2] U. Bodenhofer. The construction of ordering–based modifiers. In *Fuzzy–Neuro Systems '99*, Computational Intelligence, pages 55–62, Leipzig, 1999. Leipziger Universitätsverlag.

[3] D.E. Goldberg. *Genetic Algorithms in Search Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

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

[5] D.L.B. Jupp. The 'lethargy' theorem - a property of approximation by γ-polynomials. *Journal of Approximation Theory*, 14:204–217, August 1975.

[6] W. Lipski. Finding a manhattan path and related problems. *Networks*, 13:399–409, 1983.

[7] E. H. Mamdani. Twenty years of fuzzy control: Experiences gained and lessons learned. *IEEE International Conference on Fuzzy Systems*, pages 339–344, 1993.

[8] S. Mitaim and B. Kosko. What is the best shape of a fuzzy set in function approximation. In *IEEE International Conference on Fuzzy Systems*, pages 1237–1243, 1996.

[9] V. Soltan and A. Gorpinevich. Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. *Discret & Computational Geometry*, 9:57–79, 1993.

[10] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modelling and control. *IEEE Transactions on System, Man and Cybernetics*, SMC-15(1):116–132, 1985.

[11] L. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, 1994.

[12] L.-X. Wang. Universal approximation by hierarchical fuzzy systems. *Fuzzy Sets and Systems*, 93:223–230, 1998.

[13] 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.

[14] J. Zhang and A. Knoll. *Integrating deliberative and reactive strategies via fuzzy modular control*, chapter 15, pages 367–387. In “Fuzzy logic techniques for autonomous vehicle navigation”, edited by A. Saffiotti and D. Driankov, Springer, 2000.