A Rapid Learning Approach for Developing Fuzzy Controllers

Jianwei Zhang, Khac Van Le and Alois Knoll
Faculty of Technology, University of Bielefeld,
33501 Bielefeld, Germany
Phone: ++49-(0)521-106-2951/2952
Fax: ++49-(0)521-106-2962
E-mail: zhang|ilkhacva|knoll@techfak.uni-bielefeld.de

Abstract: In this paper we propose an approach for rapid learning an important type of fuzzy controllers. To specify linguistic terms, the B-spline basis functions are used for input variables and fuzzy singletons for output variables. "Product" is chosen as the fuzzy-conjunction, and "centroid" as the defuzzification method. By appropriately designing the rule base, a fuzzy logic controller can be interpreted as a B-spline interpolator. Such a fuzzy controller can learn to approximate any known data sequences and to minimise a certain cost functions. By choosing a suitable cost function, the learning process can converge rapidly. We present some applications of this approach in supervised learning, especially for function approximation. The approach can also be extended to the problem of unsupervised learning.

Keywords: fuzzy control, learning, B-splines.

1 Introduction

Classical Mamdani-type fuzzy controllers have been applied to diverse control problems, [3, 5]. Studying these applications, one will easily notice that the automatic design of an optimal controller becomes a very important topic, as pointed out in [1]. Recently, Sugeno type fuzzy controller has been used for function approximation and supervised learning, [4], [2]. However, a general Sugeno model cannot show the advantage of linguistic control since the higher order of polynomial combination of input variables cannot be easily extracted from the expert intuitive knowledge.

In principle, the evaluation of a fuzzy rule base is an interpolation process. Therefore, if we are considering an automatic method for designing a fuzzy controller, it is meaningful to check the interpolation methods using analytical functions. Langrage polynomials supply a set of functions which can be used for blending a given number of data points. Newton polynomials can realise the same task but they can be recursively computed so that a new polynomial does not need to be totally re-calculated for a new data. Bernstein functions are based on a parameterised data set and can also interpolate data quite well.

However, only B-spline basis functions are independent of the number of interpolation data. By choosing suitable B-spline model, the basis functions can be used as a powerful and convenient tool to specify linguistic terms.

In our previous work [6], we compared splines and a fuzzy controller with SISO (*single-input-single-output*) and MISO (*multi-input-single-output*) structures; periodical non-uniform B-spline basis functions (NUBS) are interpreted as membership functions. In this paper, we discuss the learning aspects of such a fuzzy controller and present some implemented examples.

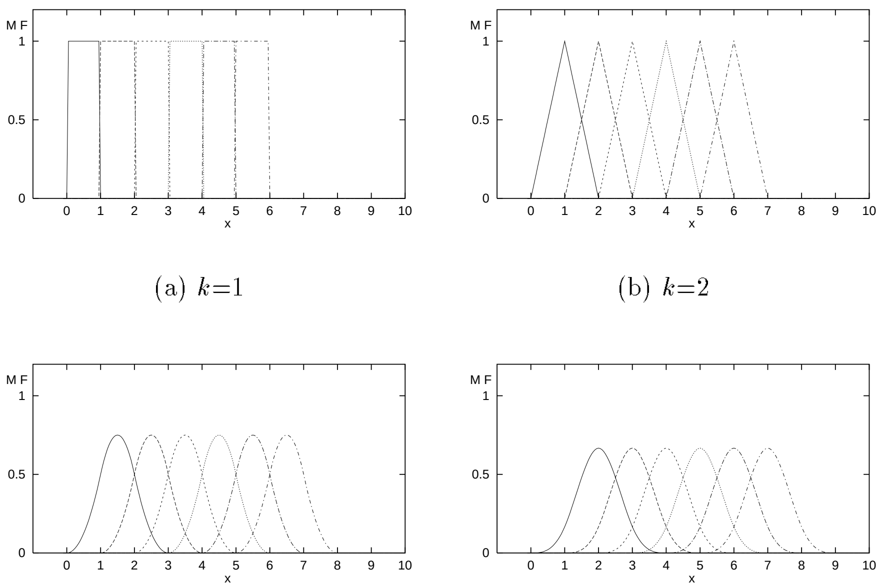
2 Constructing Fuzzy Controllers with B-Splines

2.1 B-Spline Basis Functions

Assume x is a general input variable of a control system which is defined on the universe of discourse $[x_0, x_m]$. Given a sequence of ordered parameters (knots): $(x_0, x_1, x_2, \ldots, x_m)$, the $i-$th normalised B-spline basis function (B-function) $N_{i,k}$ of order k is defined as:

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

with i = 0, 1, ... , m-k. Fig. 1 shows B-functions from order one to order 4 which can be easily associated with the fuzzy membership functions (MFs).



(a) k=1 (b) k=2

(c) k=3 (d) k=4

Figure 1: Fuzzy sets defined by B-spline basis functions of different orders.

2.2 The General MISO Controllers

Since a MIMO rule base is normally divided into several MISO rule bases, we consider only the MISO case. Under the following conditions (see [6]): a). periodical B-spline basis functions as membership functions for inputs, b). fuzzy singletons as membership functions for outputs, c). "product" as fuzzy conjunctions, d). "centroid" as defuzzification method, e). addition of "virtual linguistic terms" at both ends of each input variable and f). extension of the rule base for the "virtual linguistic terms" by copying the output values of the "nearest" neighbourhood, the computation of the output of such a fuzzy controller is equivalent to that of a *general B-spline hypersurface*. Generally, we consider a MISO system with n inputs $x_1, x_2, \ldots, x_n$, rules with the n conjunctive terms in the premise are given in the following form:

$\{Rule(i_1, i_2, \ldots, i_n)$: IF ($x_1$ is $N^1_{i_1,k_1}$) and (x_2 is $N^2_{i_2,k_2}$) and ... and (x_n is $N^n_{i_n,k_n}$)

THEN y is $Y_{i_1 i_2 \ldots i_n}\}$,

where x_j is the j-th input ($j = 1, \ldots, n$), k_j is the order of the B-spline basis functions used for x_j, $N^j_{i_j,k_j}$ is the i-th linguistic term of x_j defined by B-spline basis functions, $i_j = 0, \ldots, m_j$ represents how fine the j-th input is fuzzy partitioned, $Y_{i_1 i_2 \ldots i_n}$ is the control vertex (deBoor points) of $Rule(i_1, i_2, \ldots, i_n)$.

Then, the output y of a MISO fuzzy controller is: $y = \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j))$. This is a *general NUBS hypersurface*.

3 Learning of B-spline Fuzzy Controllers

3.1 Optimisation of the Control Vertices

A fuzzy system constructed with the approach above can be optimised with a gradient descent approach. Assume that $\{\mathbf{X}^i,\ y^i_d\}$ is a set of training data, where $\mathbf{X}^i = (x^i_1, x^i_2, \ldots, x^i_n)$ is the i-th input vector, and y^i_d is the desired output for $\mathbf{X}^i$, y^i is the output value computed by the fuzzy system. If we define the following error function: $E = \frac{1}{2} \cdot (y^i - y^i_d)^2$, the derivation of each control vertex $Y_{i_1,\ldots,i_n}$ is:

$$\Delta Y_{i_1,\ldots,i_n} = -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} = \epsilon (y^i - y^i_{soll}) \cdot \prod_{j=1}^{n} N_{i_j,k_j}(x_j)$$

where $0 < \epsilon \leq 1$. Since the second partial differentiation to $Y_{i_1,i_2,\ldots,i_n}$ is always positive, the error function E is convex. Therefore, the gradient descent approach guarantees the fast finding of the global minimum of E.

3.2 Acceleration of Rule Evaluation

The index coding of the B-functions makes the evaluation of fuzzy rules highly efficient. For an input $x \in [x_i, x_{i+1}]$, it is known that exact k linguistic terms will be activated, i.e. the values of k B-functions $N_{i,k}, N_{i-1,k}, \ldots, N_{i-k+1,k}$ are greater than zero. All the other linguistic terms are unactivated. In the whole rule base, exact $k \times n$ rules are firing for any input vector in the universe of discourse. This property makes it possible that a rule base, in which the number of rules is exponential to n, can be evaluated in a linear time with n.

3.3 Steps for Developing a B-Spline Fuzzy Controller

The steps for developing a fuzzy controller with B-spline models can be summarised as follows (M: manually, A: automatically):

1.	(M)	Select inputs.
2.	(M)	Select the order of the B-functions for each input variable.
3.	(M)	Determine the knots for partitioning each input variable.
4.	(A)	Compute the virtual and real linguistic terms for all inputs.
5.	(M/A)	Initialise the control vertices for the output.
6.	(A)	Learn the control vertices.
7.	(M/A)	If the results are satisfied, terminate.
8.	(M/A)	Modify the knots for input, go to 4; or Refine the granularity and use more training data, go to 3; or Increase the order of B-functions, go to 3; or Delete certain inputs and/or add new ones, go to 2.

In the step 3, it is very important to know how the knots should be placed in the input space. An intuitive answer is to fix the knots where the output has its extrema. If such information is available, e.g. by approximating an analytically representable function, we can apply this principle to select the knots. If the outputs of a control system are unknown, the knots can be computed with an approach similar with the optimisation of a self-organising neural network.

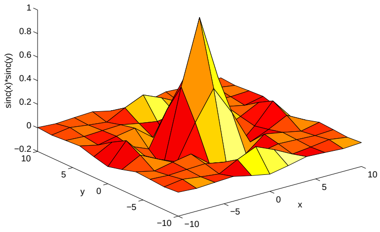
The control vertices can be initialised with the approximate *a priori* values, e.g. the experience data from experts if available. Otherwise they can be just set to zero.

4 Implementations

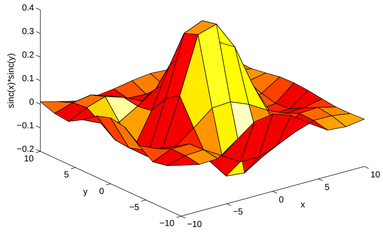
Examples in [2] were first implemented to demonstrate the learning ability for function approximation and system identification. In the following figures, the modelling of the linguistic terms using B-functions are not extra shown. Instead we use $a_1 \times a_2 \times a_3 \times \ldots$ to describe how many linguistic terms (real + virtual) are applied for the input variable $x_1, x_2, x_3, \ldots$ of the used fuzzy controller. The main focus of our observation lies on the learning process of the input-output relation, i.e. the control surface, the approximation error, and the test error.

4.1 Approximation of a Function with Two Variables $z = \frac{\sin(x)}{x} * \frac{\sin(y)}{y}$

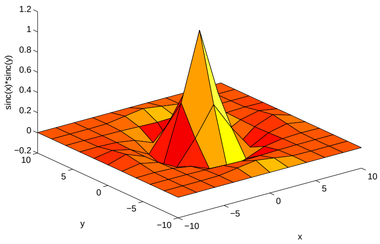
The training data were uniformly selected from the area [-10,10]$\times$[-10,10], altogether 121 training vectors $((x, y), z)$ were obtained. Since the this function is symmetric, the two input variables are covered with the same number of linguistic terms of order 3. Fig. 2(a)-(d) illustrate the training results. It can be found that relative more linguistic terms are needed for this approximation task (a 5×5 fuzzy controller even fails, Fig. 2(b)).



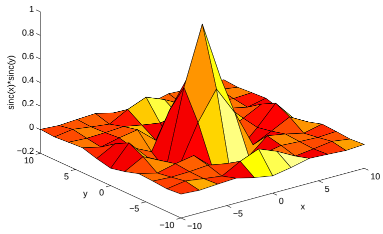
(a) Training data



(b) Output of a 5×5 controller after 1000 steps



(c) Output of a 7×7 controller after 100 steps

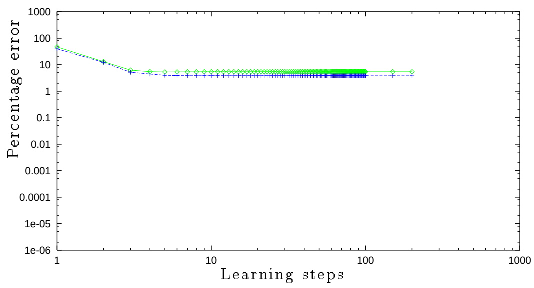


(d) Output of a 11×11 controller after 20 steps

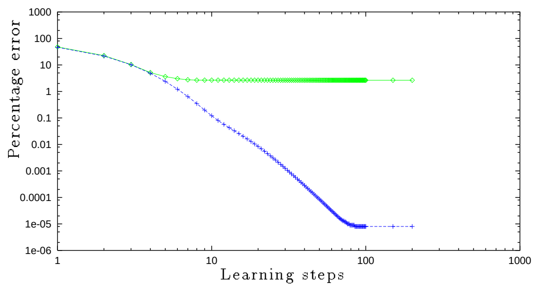
Figure 2: Approximation of the function $z = sin(x)/x * sin(y)/y$.

4.2 A Function with Three Variables $f(x, y, z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2$

216 training data and 125 test data are uniformly selected from the input space $[1, 6] \times [1, 6] \times [1, 6]$ and $[1.5, 5.5] \times [1.5, 5.5] \times [1.5, 5.5]$, respectively. Fuzzy controllers with B-functions of order 3 are trained for this example, and all the control vertices of them are initialised as zero. The percentage training and test errors are computed with the definition in [2]. In Fig. 3, the upper curve ($\diamond$) depicts the test error, the lower curve ($+$) the training error. In case of Fig. 3(b), the test error can be reduced by suing more training data.



(a) Using a 5×5×5 fuzzy controller



(b) Using a 7×7×7 fuzzy controller

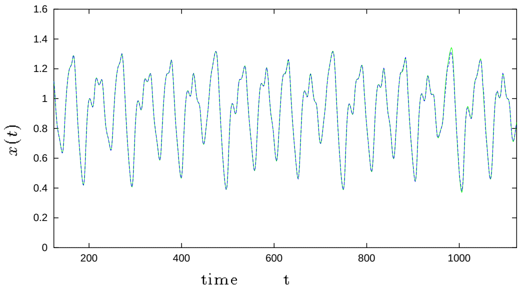
Figure 3: The error curves of two fuzzy controllers with different partition granularity.

4.3 Prediction of a Chaotic System

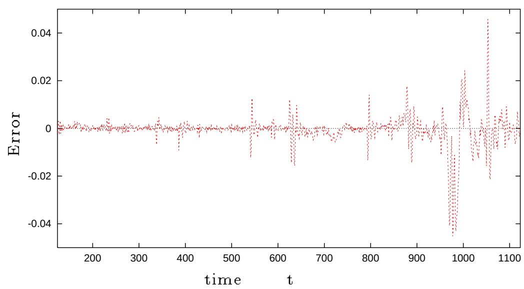
A Chaos time system is generated with the following discrete Mackey-Glass equation:

$$x(t+1) = \frac{0.2x(t-r)}{1+x^{10}(t-r)} + 0.9x(t)$$

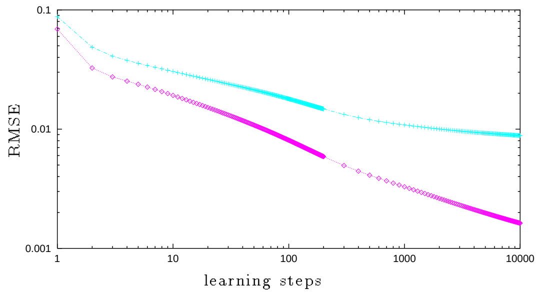
1000 training data are selected using the method in [2]. We also use 500 data as training data, other 500 as test data. The fuzzy controller has four inputs. The RMSE (Root Mean Square Error) measures the approximation error. The results are shown in Fig. 4.



(a) Output of the trained controller



(b) Approximation error



(c) RMSE / Learning steps

Figure 4: Emulation of chaotic system with a 5×5×5×5 fuzzy controller.

For this problem, the ANFIS approach [2] needs 1.5 hours on a HP Apollo 700, while our algorithm only needs 40 minutes on a SUN Sparc-4 workstation. In [2], two linguistic terms are used for each input. The training error after 500 epochs is 0.0016, the test error is 0.0015.

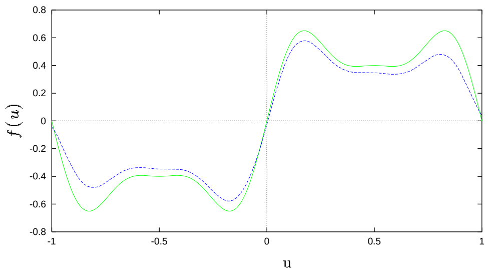
4.4 Identification of a Non-Linear System

The task is to identify a nonlinear component in a dynamic system: $y(t+1) = 0.3y(t) + 0.6y(t-1) + f(u(t))$, where $y(t)$ is the output of moment t and $u(t)$ is the input.

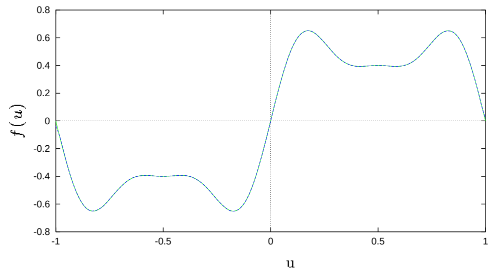
In this simulation $f(.)$ has the the form: $f(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u)$. The input signal is:

$$u(t) = \begin{cases} \sin(2\pi t/250), & \text{for } t \leq 500, \\ 0.5\sin(2\pi t/250) + 0.5\sin(2\pi t/25), & \text{for } t > 500. \end{cases}$$

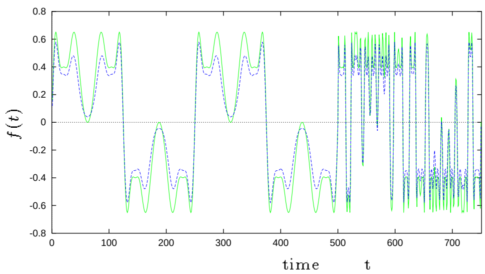
B-spline fuzzy controllers with order 3 are trained with 250 data from $t = 1$ to $t = 250$. The data from $t = 251$ to $t = 750$ are used for test. Fig. 5 shows the results with 30 B-functions after one and ten times training steps. In Fig. 5(a), (c), (e), the lighter curves represent the desired data, while in Fig. 5(b), (d), (f), these curves are well approximated.



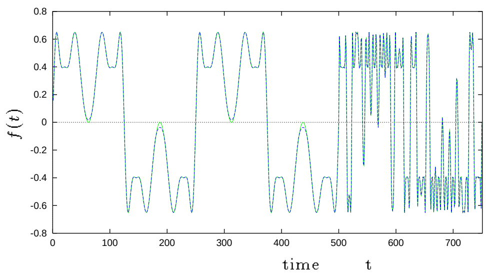
(a) The non-linear component f(u) after 1 learning step



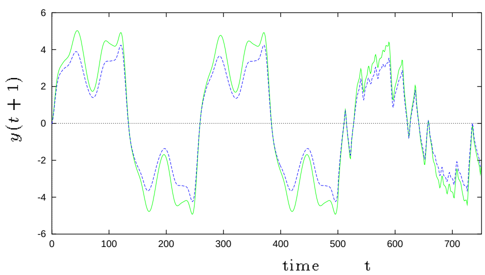
(b) f(u) after 10 learning step



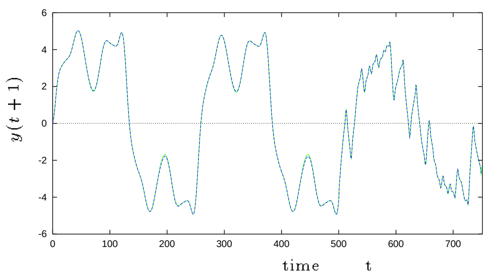
(c) f(t) after 1 learning step



(d) f(t) after 10 learning steps



(e) System output after 1 learning step



(f) System output after 10 learning steps

Figure 5: Results of the fuzzy controller with 30 B-function after learning.

5 Discussions

5.1 Supervised and Unsupervised Learning

The proposed approach is suitable for on-line learning thanks to its learning speed. We successfully applied it to the supervised learning to the "truck backer-upper" (see also

[4]) and "inverse kinematics" problem in the area of robotics. The learning approach can also be generalised for unsupervised learning by designing a suitable *evaluation function* and using such information to modify the control vertices. We have applied this approach in the mobile robot control and the sensor-based assembly operations. For further details see [8], [7]. Our current work is on extending this approach to the learning problem of large systems with multiple outputs.

5.2 Summary

We proposed a novel approach for constructing fuzzy controllers with B-spline basis functions and learning the control vertices for the THEN-parts. If the rule table is complete, then by adding certain more marginal rules, the smoothness of the controller output can be achieved by selecting the proper order of basis functions. B-spline fuzzy controllers are exact, that means no information is lost after the defuzzification. Although the number of control vertices to be optimised can be quite large in our approach, the learning process of such a fuzzy controller converges rapidly, especially for the supervised learning thanks to the one-minimum property of the error function and the efficient evaluation of the rule base. Therefore, the computation time is clearly reduced in comparison with the other approaches.

References

[1] D. Driankov, H. Hellendoorn, and R. Palm. *Some Research Directions in Fuzzy Control*, chapter 11, pages 281–312. In "Theoretical Aspects of Fuzzy Control", edited by H. T. Nyuen, M. Sugeno, R. Tong, and R. R. Yager, John Wiley & Sons, New York, 1995.

[2] J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on System, Man and Cybernetics*, 23(3):665–685, 1993.

[3] T. Terano, K. Asai, and M. Sugeno. *Applied Fuzzy Systems*. AP Professional, 1994.

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

[5] J. Yen, R. Langari, and L. A. Zadeh. *Industrial Applications of Fuzzy Logic and Intelligent Systems*. IEEE Press, 1994.

[6] J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models. In *IEEE International Conference on Fuzzy Systems*, 1996.

[7] J. Zhang, K. V. Lee, and A. Knoll. Unsupervised learning of control spaces based on b-spline models. Submitted to IEEE International Conference on Fuzzy Systems, 1997.

[8] J. Zhang, Yorck v. Collani, and A. Knoll. On-line learning of sensor-based control for acquiring assembly skills. Submitted to IEEE International Conference on Robotics and Automation, 1997.