

# FUZZY RULE BASED MODELING AS A UNIVERSAL APPROXIMATION TOOL

V. KREINOVICH

*Department of Computer Science  
University of Texas at El Paso  
El Paso, TX 79968, USA  
email vladik@cs.utep.edu*

G.C. MOUZOURIS

*Signal and Image Processing Institute  
Department of Electrical Engineering — Systems  
University of Southern California  
Los Angeles, CA 90089-2564, USA  
email mouzouri@dorrit.usc.edu*

AND

H.T. NGUYEN

*Department of Mathematical Sciences  
New Mexico State University  
Las Cruces, NM 88003-8001, USA  
email hunguyen@nmsu.edu*

## 1. INTRODUCTION

### 1.1. WHY UNIVERSAL APPROXIMATION

**In some cases, fuzzy rule based model needs tuning.** If we have applied some version of fuzzy rule based modeling, and the resulting model is satisfactory, great. But sometimes, the resulting model is not of very high quality:

- we may have misinterpreted some of the expert's rules;
- we may have missed some of the important rules.

So, to improve the quality of the resulting model, we must either:

- re-interpret the existing rules, or
- ask the experts for extra rules.

**Before we start tuning, we want to make sure that tuning will help.** In order to make sure that this “tuning” will always help, we must guarantee that by appropriate tuning, we will be able to change the initial (not very successful) model into a successful one.

To guarantee that the fuzzy rule based modeling methodology will work in all possible situations, we must make sure that *every possible system can be obtained by applying the fuzzy rule based modeling methodology to appropriate rules.*

**Measurements are never 100% accurate, hence, it is sufficient to approximate the desired system.** Realistically, measurements are never absolutely precise, hence, we do not need to model the system *exactly*; it is quite sufficient to be able to *approximate* the system, within the measurement accuracy  $\varepsilon$ , by using appropriate if-then rules.

**Mathematical reformulation of the desired property: universal approximation.** In more mathematical terms, we want to guarantee that, given an arbitrary function  $u(x_1, \dots, x_n)$ , and an arbitrary positive real number  $\varepsilon > 0$ , we will be able to find a function  $\tilde{u}(x_1, \dots, x_n)$  generated by fuzzy rule based modeling methodology that is  $\varepsilon$ -close to the original function  $u(x_1, \dots, x_n)$ . In mathematics, this ability of approximate an arbitrary function with an arbitrary accuracy is called a *universal approximation* property.

In this chapter, we will briefly review the universal approximation results for fuzzy rule based modeling methodology. This chapter is *not* intended as an exhaustive survey of different universal approximation *results*; our (slightly less ambitious) goal was to overview all possible *approaches*. Because of this, our bibliography is far from being complete.

Also, this chapter is intended to be more an exposition of *ideas*, rather than technical details; the readers who are interested in technical details should refer to the papers that we cite in this survey.

The paper is structured as follows: we start with the main universal approximation results. Then, we describe auxiliary approximation results; in particular, we analyze when it can be guaranteed that the approximating model has the desired properties (such as smoothness). Other auxiliary universal approximation results include:

- systems that use additional expert information (e.g., if-then rules that describe the system and if-then rules that describe the control);
- systems that use unusual logical connectives; and
- approximation results that take the learning process into consideration (i.e., mainly, for fuzzy neural networks).

These universal approximation results are theoretically valid, but these results do not always lead to a practical fuzzy model:

- they do not take into consideration inevitable inaccuracy in the input data, and, besides,
- they may require unrealistically many rules.

So, next, we describe the known method of making the approximation results more realistic.

In this chapter, we describe many different variants of fuzzy rule based modeling methodology. The natural question is: which of these variants should we use? One natural requirements is that the chosen variant must have a universal approximation property. However, since practically all variants have this property, this requirement need to be supplemented by the comparison of the *quality* of the corresponding approximations. We can also go one step back and ask: when should we use fuzzy rule based modeling methodology at all, and when are other intelligent modeling methodologies (such as neural network modeling) better? The answers to these comparison questions is given in the last two sections of this chapter.

## 2. MAIN UNIVERSAL APPROXIMATION RESULTS

### 2.1. UNIVERSAL APPROXIMATION PROPERTY FOR THE ORIGINAL MAMDANI APPROACH

Before we describe the universal approximation results, let us briefly recall the main formulas of fuzzy rule based modeling methodology (see Chapter 1 for the detailed description).

**Rules.** Fuzzy rule-based modeling methodology starts with expert “if-then” rules, i.e., with rules of the following type:

If  $x_1$  is  $A_1^j$  and  $x_2$  is  $A_2^j$  and ... and  $x_n$  is  $A_n^j$ , then  $u$  is  $B^j$ ,

where  $x_i$  are parameters that characterize the system’s input,  $u$  is the output, and  $A_i^j$ ,  $B^j$  are the natural language terms that are used to describe the  $j^{\text{th}}$  rule (e.g., “small”, “medium”, etc).

*Comment.* One of the main applications of fuzzy rule based modeling methodology is *fuzzy control*, when we want to simulate the control applied by expert human controllers. In such control applications, the output  $u$  describes applied control.

**Mamdani’s transformation.** The value  $u$  is a proper value for the output if and only if one of these rules is applicable.

Thus, the property “ $u$  is a proper output” (which we will denote by  $C(u)$ ), can be, therefore, described as follows:

$$\begin{aligned}
C(u) \equiv & (A_1^1(x_1) \& A_2^1(x_2) \& \dots \& A_n^1(x_n) \& B^1(u)) \vee \\
& (A_1^2(x_1) \& A_2^2(x_2) \& \dots \& A_n^2(x_n) \& B^2(u)) \vee \\
& \dots \\
& (A_1^K(x_1) \& A_2^K(x_2) \& \dots \& A_n^K(x_n) \& B^K(u))
\end{aligned}$$

**“And” and “or” operations.** The natural language terms are described by *membership functions*, i.e., we describe  $A_i^j(x)$  as  $\mu_i^j(x) \in [0, 1]$ , and  $B^j(u)$  as  $\mu_j(u)$ .

The logical connectives  $\&$  and  $\vee$  are interpreted, in this context, as operations  $f_{\&}$  and  $f_{\vee}$  on truth values. The most frequent choices of these operations are  $\min(a, b)$  and  $a \cdot b$  for  $f_{\&}(a, b)$ , and  $\max(a, b)$  and  $a + b - a \cdot b$  for  $f_{\vee}(a, b)$ <sup>1</sup>.

After these interpretations, we can form the membership function for the output:

$$\mu_C(u) = f_{\vee}(p_1, \dots, p_K),$$

where

$$p_j = f_{\&}(\mu_{j,1}(x_1), \mu_{j,2}(x_2), \dots, \mu_{j,n}(x_n), \mu_j(u)).$$

**Defuzzification.** The model must return a single output value  $u$ . An operation that transforms a membership function into a single value is called a *defuzzification*. To complete the fuzzy rule based modeling methodology, therefore, we must apply some defuzzification operator  $F$  to the membership function  $\mu_C(u)$  and thus obtain the crisp output value  $\bar{u} = f_C(x)$  that corresponds to  $x = (x_1, \dots, x_n)$ . The most widely used defuzzification procedure is *centroid defuzzification*

$$\bar{u} = \frac{\int u \cdot \mu_C(u) du}{\int \mu_C(u) du}.$$

The above formulas can be further simplified if we only allow fuzziness in the *inputs*, and require the outputs of each rule to be crisp (i.e., to be numbers). In other words, we only allow rules of the type:

If  $x_1$  is  $A_1^j$  and  $x_2$  is  $A_2^j$  and  $\dots$  and  $x_n$  is  $A_n^j$ , then  $u = u_j$ .

For such rules, the above methodology leads to

$$\bar{u} = \frac{\sum r_j \cdot u_j}{\sum r_j},$$

<sup>1</sup>Sometimes, the term “Mamdani approach” is used only for the case when  $f_{\&}(a, b) = \min(a, b)$ . The use of  $f_{\&}(a, b) = a \cdot b$  is called *Larsen* approach, after the author of the paper [1] that described the first successful applications of this “and” operation.

where

$$r_j = f_{\&}(\mu_{j,1}(x_1), \mu_{j,2}(x_2), \dots, \mu_{j,n}(x_n)).$$

**Who was the first.** As successful fuzzy models and fuzzy controls were found for more and more systems, the belief grew that fuzzy rule based modeling methodology is indeed a universal approximation tool.

This belief was explicitly formulated, e.g., by the authors of [18] who showed experimentally that an arbitrary function serving as a control strategy can be well approximated by fuzzy controllers.

Several different universal approximation results expressing this belief were formulated and proved, almost simultaneously, in 1990–92 papers by J. Buckley, Z. Cao, E. Czogala, D. Dubois, M. Grabisch, J. Han, Y. Hayashi, C.-C. Jou, A. Kandel, B. Kosko, J. Mendel, H. Prade, and L.-X. Wang<sup>2</sup>. In the following text, we will describe what each of these authors proved.

**Universal approximation result for arbitrary inputs.** Bart Kosko, in [20], analyzed the most general case of fuzzy rule based modeling, in which the input  $x$  is not necessarily a finite sequence of real numbers (i.e., not necessarily  $x \in R^n$ ), but an element of an arbitrary *compact* set  $X$ . He has shown (see also [19], [21], [22], [23], [24], [26], [25]) that for an appropriate choice of an aggregation ( $\vee$ ) operation (namely, for  $f_{\vee} = +$ ), and for  $f_{\&}(a, b) = a \cdot b$ , an arbitrary continuous function  $f : X \rightarrow R$  defined on an arbitrary compact  $X$  can be approximated by functions that result from fuzzy if-then rules of the type “if  $A(x)$  then  $B(u)$ ”, where  $x \in X$ ,  $A(x)$  is a fuzzy property (fuzzy subset) of a compact  $X$ , and  $B(u)$  is a fuzzy property of real numbers.

The choice of  $+$  as “or” and  $\cdot$  as “and” makes computations simpler: Indeed, since  $\mu_C$  is the sum and  $p_j$  is the product ( $p_j = \mu_{j,1}(x) \cdot \mu_j(u)$ ), the integrals in Mamdani’s approach are simplified into

$$\int u \cdot \mu_C(u) du = \int u \cdot \sum(\mu_{j,1}(x) \cdot \mu_j(u)) du = \sum r_j \cdot U_j,$$

where  $r_j = \mu_{j,1}(x)$  and  $U_j = \int u \cdot \mu_j(u) du$ , and similarly,

$$\int \mu_C(u) du = \int \sum(\mu_{j,1}(x) \cdot \mu_j(u)) du = \sum r_j \cdot V_j,$$

where  $V_j = \int \mu_j(u) du$ . Therefore, the Mamdani approach formula is simplified into

$$\bar{u} = \frac{\sum r_j \cdot U_j}{\sum r_j \cdot V_j}.$$

<sup>2</sup>The authors of these papers are listed in *alphabetic*, not in chronological order.

**Towards a more realistic universal approximation result.** For the practical case of several inputs  $x_1, \dots, x_n$ , the compact  $X$  is a subset of an  $n$ -dimensional space. In this case, Kosko's original theorem requires rules "if  $A$  then  $B$ " with arbitrary fuzzy properties  $A(x_1, \dots, x_n)$ . In the standard fuzzy rule based modeling technique, however, the conditions of the rules are usually of the type  $A_1(x) \& \dots \& A_n(x_n)$ . Are rules with such conditions sufficient?

In [31], [29], [32], [33], [30], it was shown that if we use *Gaussian* membership functions, product fuzzy conjunction, and center of average defuzzification, we get a universal approximation<sup>3</sup>.

In [27], [28], it was shown that this universal approximation property holds for an *arbitrary* shape of membership functions and for arbitrary "and" and "or" operations, and for an arbitrary defuzzification procedure (see also [86], [213]).

Different universal approximation results were also proven in [2], [9], [14], [15], [16], [17], [3], [4], [8], [10], [5], [7], [6], [34], [35], [36], [13], [37]. (Most of these results will be described in some detail in the following text.)

## 2.2. GENERALIZATIONS OF THE STANDARD FUZZY RULE BASED MODELING METHODOLOGY HAVE A UNIVERSAL APPROXIMATION PROPERTY

**General idea.** Since the standard fuzzy rule based modeling methodology has a universal approximation property, any *generalization* of this methodology also has this property.

Several generalizations of the standard methodology have been proposed:

- with more complicated *conclusions* of if-then rules;
- with more complicated *conditions* of if-then rules;
- with more complicated *implication*;
- that take into consideration that experts may assign *different degrees of truth* to different *rules*;
- that take into consideration that different groups of experts may formulate *inconsistent sets of rules*.

**Fuzzy rule based modeling with more complicated conclusions of if-then rules: Takagi-Sugeno models are universal approximation tools.** In the above approaches, we assumed that for each rule, the output is *fixed* (i.e., does not explicitly depend on the input). In some real-life situations, however, an expert can not only explain that, say, for small  $x_1$  the output should be small, but this expert can also explicitly describe

<sup>3</sup>The result is actually proven not only for Gaussian functions, but also for functions from any class  $\mathcal{F}$  that is closed under shift and multiplication.

a reasonable dependency of  $u$  on  $x_1$  for the case when  $x_1$  is small. This dependency  $u = f_j(x_1, \dots, x_n)$  is usually described by a linear function, but more complicated dependencies are also possible. In these situations, expert rules are of the type:

If  $x_1$  is  $A_1^j$  and  $x_2$  is  $A_2^j$  and  $\dots$  and  $x_n$  is  $A_n^j$ , then  $u = f_j(x_1, \dots, x_n)$ , and lead to the model of the form

$$\bar{u} = \frac{\sum r_j \cdot f_j(x_1, \dots, x_n)}{\sum r_j},$$

where

$$r_j = f_{\&}(\mu_{j,1}(x_1), \mu_{j,2}(x_2), \dots, \mu_{j,n}(x_n)).$$

This idea is called *Takagi-Sugeno* approach, after its authors [45] (see Chapters 2 and 7 for more detail). For this approach, the universal approximation property was proven in [38], [39], [40], [41], [44].

A natural further generalization of this approach was proposed in [42], [43]. In which in the conclusion of each rule, the desired output  $u$  is given not by an explicit formula, but by a (crisp) *dynamical system*, i.e., by a system of differential equations that determine the time derivative of the output variable (i.e., its change in time) as a function of the inputs and of the previous values of the output. This generalization also has a universal approximation property [42], [43].

**Fuzzy if-then rules with more complicated conditions: ellipsoid approach.** In traditional fuzzy rule based modeling, the condition of each rule is a *conjunction* of several conditions related to different inputs, i.e., each condition is of the type  $A_1(x_1) \& \dots \& A_n(x_n)$ . Crudely speaking, we can say that in these conditions, different inputs are *independent* in the sense that the degree with which each input  $x_i$  satisfies the condition does not depend on the values of the other inputs. In particular, when each condition  $A_i(x_i)$  is described by a Gaussian membership function, and we use product as “and”, the resulting membership function for the condition  $\mu(x) = \mu_1(x_1) \cdot \dots \cdot \mu_n(x_n)$  is also Gaussian, and its  $\alpha$ -cut (i.e., the set of all values  $x = (x_1, \dots, x_n)$  for which  $\mu(x) \geq \alpha$ ) is an *ellipsoid* with axes coinciding with the coordinate axes.

In some real-life cases, however, there *is* a dependency. As a result, some conditions  $A_i^j$  in some expert rules may describe not a single input, but a *combination* of such inputs. The simplest case is when we have a *linear combination*. In this case, each expert rule has the form

If  $x_1^{(j)}$  is  $A_1^j$  and  $x_2^{(j)}$  is  $A_2^j$  and  $\dots$  and  $x_n^{(j)}$  is  $A_n^j$ , then  $u$  is  $B^j$ ,

where  $x_i^{(j)} = w_{i1}^{(j)} \cdot x_1 + \dots + w_{in}^{(j)} \cdot x_n$  is a linear combination of the (measured) inputs  $x_i$ . If we use Gaussian membership functions and  $a \cdot b$  as “and”, then

the resulting membership function is still Gaussian, and its  $\alpha$ -cut is an *arbitrarily* oriented ellipsoid. For rules with such ellipsoid-based conditions, a universal approximation property is proven in [46], [47], [50], [51], [52], [48], [49], [53].

**Fuzzy if-then rules with a more complicated implication.** When an expert pronounces the rule “if  $x$  is  $A$ , then  $u$  is  $B$ ”, e.g., “if an obstacle is close, break”, she means not only that if  $A(x)$  is true, then  $B(u)$  should be true: it also means that if  $A(x)$  is *somewhat* true (i.e., if an obstacle is close), then  $B(u)$  should also be somewhat true (i.e., we should break some), and the closer the obstacle, the more intensely we need to break. In other words, in addition to the original rule, must we have a “gradual” rule “the more  $x$  is  $A$ , the more  $u$  is  $B$ ”.

Mamdani’s approach, in which if-then rules are interpreted by using “and”, does not always capture these (implicitly assumed) additional “gradual rule”. To capture these meanings, the authors of [54], [55], [56] propose to use a *fuzzy implication* operation  $f_{\rightarrow}(a, b)$  instead of “or”. In other words, the degree  $p_j$  with which  $j$ -th rule “if  $A^j(x)$  then  $B^j(u)$ ” is true is calculated as  $p_j = f_{\rightarrow}(r_j, \mu_j(u))$  (where  $r_j$  is the grade to which the condition  $A^j(x)$  is true), and the grade to which all  $K$  rules from the rule base are true is calculated as  $f_{\&}(p_1, \dots, p_K)$ .

In particular, if we use the simplest possible fuzzy implication  $f_{\rightarrow}(a, b)$ , that is equal to 1 if  $a \geq b$  and to 0 else, then the membership function  $\mu_C(u)$  describes a *crisp* set of possible output values  $u$  for which  $\mu_j(u) \geq r_j$  for all  $j$ .

It may happen that rules are inconsistent, and there will be no such  $u$ ; to handle this situation, special interpolation rules are designed. For the resulting methodology, the universal approximation property is proven.

**Different grade of truth assigned to different rules.** Another generalization of the standard fuzzy rule based methodology is analyzed in [60], [57], [58], [62], [59], [61]: In standard fuzzy rule based modeling methodology, all rules are considered to be equally valid. In reality, however, experts may have *more confidence in some if-then rules* and less confidence in the others. In the spirit of fuzzy approach, the “degree of confidence” (truth value) of each  $j$ -th rule can be described by a number  $w_j$  from the interval  $[0, 1]$ . This number  $w_j$  determines, crudely speaking, how much “weight” we assign to each rule; therefore, this number is also called the *weight* of  $j$ -th rule.

How can we take these weights into consideration when producing a fuzzy rule based model? In *traditional* fuzzy rule based modeling methodology, we are 100% confident in each rule. As a result, we consider a rule “If  $x$  is  $A^j$ , then  $u$  is  $B^j$ ” to be applicable if both its condition  $A^j(x)$  and

its conclusion  $B^j(u)$  hold, i.e., we take the rule's truth value to be equal to  $p_j = f_{\&}(r_j, \mu_j(u))$ , where  $r_j$  is the grade with which the fuzzy property  $A^j$  holds for the given input  $x$ . In the *new* approach, we have some doubts in this rule; as a result, the rule is applicable when not only its conditions and conclusions hold, but also when the rule itself is valid. Thus, the degree of applicability  $p_j$  of  $j$ -th rule can be obtained by applying the “and” operation to *three* statements instead of two:  $p_j = f_{\&}(w_j, r_j, \mu_j(u))$ . The resulting formulas for fuzzy rule based modeling becomes correspondingly more complicated.

In particular, when  $f_{\&}(a, b) = a \cdot b$ , we have  $p_j = w_j \cdot f_{\&}(r_j, \mu_j(u))$ ; in other words, taking the weights into consideration means that we multiply each rule's degree of applicability by this weight. If, in addition, we use + as “or”, we get the following formula  $\bar{u} = (\sum w_j \cdot r_j \cdot U_j) / (\sum w_j \cdot V_j)$ . Even this particular case of “weighted” fuzzy rule based modeling has a universal approximation property.

**Several different (inconsistent) sets of rules.** Yet another generalization of the standard fuzzy rule based modeling methodology was proposed and analyzed by J. Buckley, B. Kosko, *et al.* Fuzzy rule based modeling methodology starts with a single set of expert rules. In some situations, however, experts cannot agree on the rules. In such situations, we have several different sets of rules; these sets of rules are incompatible and therefore, cannot be easily merged together. Hence, we have to apply fuzzy rule based modeling methodology to each set of rules. As a result, we get several different models  $\bar{u}^{(1)}(x_1, \dots, x_n), \bar{u}^{(2)}(x_1, \dots, x_n), \dots$ . A (more traditional) approach would be to choose *one* of these sets of rules, based on the quality of the resulting models, and simply disregard all the other sets of rules, i.e., to choose  $\bar{u} = \bar{u}^{(k_0)}$ , where  $k_0$  is the best model. However, other sets of rules were also proposed by *experts*; this means that they may not be the best *always*, but they definitely describe some expert knowledge that we would want to use in our model. So, it is desirable to take these other models into consideration: e.g., instead of *selecting* the “best” of the models that correspond to different sets of rules (i.e., instead of choosing  $\bar{u} = \bar{u}^{(k_0)}$ ), we can *combine* the values  $\bar{u}^{(k)}$  produced by the different models, e.g., into a linear combination  $\bar{u} = \sum w^{(k)} \cdot \bar{u}^{(k)}$ . In this combination, the weight  $w^{(k)}$  of a given model is proportional to the relative quality of corresponding the set of if-then rules: the better the rules, the larger the weight. Such “hierarchical” models are described and analyzed in [65], [68], [69], [63], [64], [66], [70], [67]. Since the resulting model is a generalization of the standard fuzzy rule-based models (namely, the standard rule-based model corresponds to the case when we have only one set of rules), these models also have the universal approximation property.

The resulting combined model is, still, not always of the ideal qual-

ity. Ideally, to handle this situation, we should *go back all the way* to the rules, ask the experts to reconcile their rules, and then process the resulting *consistent* combined set of rules. Since this reconciliation is not always possible, the paper [71] proposes the second best thing: *go back one step*, and, instead of combining the *values*  $\bar{u}^{(k)}$  that come from different sets of rules, combine the corresponding *membership functions*  $\mu_C^{(k)}(u)$ , and then apply defuzzification to the combined membership function  $\mu_C(u)$ . Such a combination often leads (not surprisingly) to a better quality model than the combination of values from different models.

### 2.3. SIMPLIFIED VERSIONS OF THE STANDARD FUZZY RULE BASED MODELING METHODOLOGY ALSO HAVE A UNIVERSAL APPROXIMATION PROPERTY

In the original Mamdani approach, all the stages before defuzzification involve the explicit application of functions, while defuzzification includes *integration* and is, therefore, the most computationally complicated (and thus, time-consuming) stage. For general membership functions, we have to use the general method of computing the integrals: numerical differentiation. For time-sensitive applications such as on-line control, the implementation of this formula can take too much time.

We can, of course, *pre-compute* the values of the model for all possible combinations of inputs. In this case, to find the desired value, we simply need to look up the value of  $\bar{u}$  in the corresponding place of the pre-computed *look-up table*. Look-up tables save some time, but require lots of computer memory to store, and, besides, if a table is large, looking up can still take too much time.

For the most widely used membership functions — triangular and trapezoidal — we can find explicit analytical formulas for the integral, and use these formulas instead of the numerical integration [74], [75]. The use of these analytical formulas saves time, but in some practical situations, the resulting formulas may still take too long to compute.

The simplified Mamdani approach is easier to compute, but for some problems, it can still take too much time. For such problems, it is therefore desirable to further simplify this formula. There have been two simplification proposals:

- The simplified Mamdani's defuzzification formula contains several multiplications, additions, and a division. The most time-consuming of them is division. So, to maximally save time, we can *eliminate division*. As a result, we get a formula  $\bar{u} = \sum r_j \cdot u_j$  (for Mamdani approach) or  $\bar{u} = \sum r_j \cdot f_j(x_1, \dots, x_n)$  (for Takagi-Sugeno approach). These formulas were proposed in [72], [73], [76], [77]. For Takagi-Sugeno version,

with polynomial  $f_j$ , the universal approximation property was proven in [72]. In [76], [77], it is shown that we still get the universal approximation property even if we use Mamdani approach (i.e., use constants  $u_j$  instead of arbitrary functions  $f_j(x_1, \dots, x_n)$ ).

- With or without division, the simplified Mamdani approach returns the value  $\bar{u}$  that is a linear combination of the values  $u_j$  corresponding to different rules. When we use division, we *normalize* this linear combination and thus, make sure that the resulting value  $\bar{u}$  is in between the smallest and the largest of the values  $u_j$ . Eliminating division makes the result fast to compute, but the downside is that since we eliminated the normalization, the resulting value  $\bar{u}$  can be way off, i.e., it can be much larger (or much smaller) than any of the values  $u_j$  recommended by the expert rules. So, if we want to value  $\bar{u}$  to stay within the reasonable bounds, we must keep a normalization step, and thus, *keep division*.

In this case, the only way to save computation time is to *save on* two other operations: *multiplication and addition*. The only computer operations with real numbers that are faster than the basic arithmetic operations (addition and multiplication) are minimum and maximum. Therefore, the authors of [78] recommend using max using min instead of multiplication and max instead of addition (from the viewpoint of fuzzy logic, this is a very natural choice, because both max and + are the major examples of “or” operations, and min and product are the main examples of “and” operations). In other words, instead of  $\bar{u} = (\sum r_j \cdot u_j) / (\sum r_j)$ , the authors of [78] suggest using  $\bar{u} = (\max \min(r_j, u_j)) / (\max r_j)$ ; they show that fuzzy rule based modeling with thus simplified defuzzification rule also has a universal approximation property.

- Another possibility to avoid division is to use *Mean of Maximum* defuzzification, i.e., take as the desired value  $\bar{u}$  either the value for which  $\mu_C(u) \rightarrow \max$ , or, if there are several such values, the midpoint of the interval of such values  $u$ . The universal approximation property for this defuzzification is proven in [79].

#### 2.4. FUZZY RULE-BASED SYSTEMS ARE UNIVERSAL APPROXIMATION TOOLS FOR DISTRIBUTED SYSTEMS

**Distributed systems.** In the previous section, we explained that fuzzy rule-based systems are universal approximation tools for systems that can be describe by finitely many parameters  $x_1, \dots, x_n$ . However, not all real-life systems can be described by finitely many parameters. For example, an appropriate description of a *chemical reaction* requires knowledge of

the temperature  $T$ , density  $\rho$ , and other characteristics at *all* the points  $\vec{u}$  inside the reactor. In this case, to describe a state of the plant, we must know the *functions*  $T(\vec{u})$ ,  $\rho(\vec{u})$ , etc. If we are describing (and/or controlling) a plant which consists of several reactors, we need to know the functions  $T(\vec{u})$ ,  $\rho(\vec{u})$ ,  $\dots$ , for the first reactor  $U_1$ , and also similar functions to describe other reactors  $U_2, \dots$ . In general, to describe a state of the system, we need several *functions*  $f_1(x), \dots, f_n(x)$  instead of several *numbers*. Such systems are called *systems with distributed parameters*, or *distributed systems*.

**Rule based description of distributed systems.** How can we formulate expert rules for such systems? In reality, during a finite period of time, we can only measure the values of a function in *finitely many* points. Therefore, e.g., the actual simulator or controller can only use *finitely many* values  $f_i(x_1^j), \dots, f_i(x_n^j)$  of each function  $f_i$ . Thus, reasonable expert rules describe what to do if we know finitely many values, i.e., they are the rules of the type

If  $f_1(x_{11}^j)$  is  $A_{11}^j$ ,  $f_1(x_{12}^j)$  is  $A_{12}^j$ ,  $\dots$ ,  $f_1(x_{1m}^j)$  is  $A_{1m}^j$ ,  $f_2(x_{21}^j)$  is  $A_{21}^j$ ,  $\dots$ ,  
and  $f_n(x_{nm}^j)$  is  $A_{nm}^j$ , then  $u$  is  $B^j$ ,

where  $f_i(x)$  are functions that characterize the state of the system,  $x_{kl}^j$  are the values in which these functions  $f_i(x)$  are measured,  $u$  is the output, and  $A_{kl}^j$ ,  $B^j$  are the natural language terms that are used to describe the  $j^{\text{th}}$  rule (e.g., “small”, “medium”, etc). For these rules, the standard fuzzy rule-based methodology leads to

$$\bar{u} = \frac{\int u \cdot \mu_C(u) du}{\int \mu_C(u) du},$$

where

$$\mu_C(u) = f_{\vee}(p_1, \dots, p_K),$$

and

$$p_j = f_{\&}(\{\mu_{kl}^j(f_k(x_{kl}^j))\}_{k,l}, \mu_j(u)).$$

A natural question is: do fuzzy rule-based systems of this type universally approximate arbitrary functions  $u(f_1, \dots, f_n)$ ? The positive answer to this question is given in [81], [82], [80], [83].

**Auxiliary result: approximation of universal controllers.** The previous results show that the fuzzy rule based methodology is a universal approximation tool. In particular, for control applications, this means that an arbitrary control strategy can be approximated, within an arbitrary accuracy, by an approximate fuzzy controller.

The previous universal approximation results address control of a *single* plant under one *specific* control objective. In real life, however, we may have

*different* plants and *different* control objectives. Therefore, the objective of control theory is not only to provide optimal control of *specific* plants under *specific* criteria, but also to develop *general* methods that would help to control an *arbitrary* plant under an *arbitrary* objective. We would, therefore, like fuzzy methodology to provide us with a *universal controller* in the following sense: we supply it with the description of the system and with an objective (i.e., an optimality criterion), and it will generate the optimal control for this very system under this very criterion.

In [81], [82], [83], it is shown that fuzzy control is a universal tool for such universal controllers as well, i.e., that for every universal controller  $s$  of this type, and for every given accuracy  $\varepsilon > 0$ , we can formulate expert rules that fuzzy control methodology will transform into a universal controller strategy  $\tilde{s}$  that is  $\varepsilon$ -close to  $s$ .

## 2.5. FUZZY RULE-BASED SYSTEMS ARE UNIVERSAL APPROXIMATION TOOLS FOR DISCRETE SYSTEMS: APPLICATION TO EXPERT SYSTEM DESIGN

### **Fuzzy control methods can be used to control computer processes.**

In the above example, we considered *continuous* systems, i.e., systems described by one or several *continuous* parameters. It turns out that the same fuzzy rule based modeling and fuzzy control methodology is useful for *discrete* systems, i.e., systems in which state variables only take discrete values.

Computers are a natural example of such discrete systems. So, in this section, we will describe how fuzzy control can be used to control computer processes.

**Expert systems.** As an example of such a computer process, we can take the process of answering queries in expert systems. An *expert system*, for a certain area of expertise, is a computer system that tries to simulate experts' answers to different questions (*queries*) about this area; e.g., a medical expert system must, given symptoms of a patient, return the possible diagnoses and a reasonable treatment. To be able to do this, an expert system must contain the expert's knowledge; this computer-stored knowledge is called a *knowledge base*.

In addition to the knowledge base, the system must contain a program for answering queries; such a program is called an *inference engine*.

**Expert systems are “universal” tools.** Designing an inference engine is a very difficult problem: Even when we have *crisp* knowledge, and the knowledge base contains only *propositional* statements  $F_i$  — i.e., statements obtained from the elementary statements  $S_1, \dots, S_n$  (like “a patient has a flu”) by using “and” (&), “or” ( $\vee$ ), and “not” ( $\neg$ ) — the question of

whether a given query follows from the knowledge  $F_1, \dots, F_m$  is, in general, computationally intractable (NP-hard) [84].

NP-hard means that this problem is *universal* in the following sense: any other problem from a very reasonable class (called *NP*) can be reduced to a particular case of this query-answering problem. This universality means that this problem is indeed very difficult to solve: if we could have an algorithm that solves all particular cases of this problem in reasonable time, then we would be thus be able to solve not only this problem, but also *all* reasonable problems. So, this query-answering problem is as computationally difficult as the most complicated of these (realistic) problems.

**Heuristic methods are needed.** NP-hard means, crudely speaking, that no algorithm can solve *all* particular cases of this problem in reasonable time; thus, *heuristic methods* are needed. In other words, we not only need expert's knowledge about the *domain* to which this expert system is applied, but we also need expert knowledge about the *way* experts answer queries.

**It is natural to use fuzzy values (between 0 and 1) to describe heuristic methods.** If we ask an expert about a certain query, this expert will often come up with a crisp answer ("yes" or "no"). Producing this answer takes some time. If we ask for an expert's opinion before this time, we will get his *preliminary* opinion; in this preliminary opinion, she is not yet sure whether the answer will be "yes" or "no", but she will probably have some *degree of belief* either in a "yes" answer, or in a "no" answer, or maybe in both. If we ask an expert for the reasons for this degree of belief, she will probably describe some beliefs in the elementary statements  $S_1, \dots, S_n$  and/or their negations.

Therefore, it is natural to simulate this expert reasoning as a step-by-step procedure, in which we start with no beliefs at all (except for the knowledge contained in the knowledge base) and then *modify* our degrees of belief  $d(S_i)$  and  $d(\neg S_i)$  in the basic statements  $S_i$  and their negations  $\neg S_i$ .

**The use of fuzzy rule based modeling methodology.** In order to apply fuzzy rule based modeling methodology, we must describe this change in degrees of belief by if-then rules.

A natural way to do this comes from the fact that our knowledge consists of propositional formulas, and it is known that every propositional formula can be reformulated in Conjunctive Normal Form (CNF), i.e., in the form  $D_1 \& \dots \& D_k$ , where each of the formulas  $D_j$  (called *disjunctions*) is of the type  $a \vee b \vee c$ , and  $a$ ,  $b$ , and  $c$  are *literals* (i.e., elementary statements  $S_i$  or their negations  $\neg S_i$ ). Each disjunction  $D_j$ , in its turn, can be reformulated as three implications: "if  $\neg a$  and  $\neg b$ , then  $c$ "; "if  $\neg b$  and  $\neg c$ , then  $a$ "; and

“if  $\neg a$  and  $\neg c$ , then  $b$ ”. Thus, the entire knowledge can be represented as a set of such if-then rules.

These rules describing *knowledge* naturally lead to the rules describing *change in degrees of belief*: e.g., the knowledge rule “if  $\neg a$  and  $\neg b$ , then  $c$ ” leads to the update rule “if  $\neg a$  and  $\neg b$ , then increase the degree of belief in  $c$ ”. Thus, we can apply the standard fuzzy rule based modeling methodology to these rules: at any given moment of time, we know the degree of belief  $d(\neg a)$  in  $\neg a$  and the degree of belief  $d(\neg b)$  in  $\neg b$ . Therefore, we can compute the degree of belief  $p_j$  that this particular rule is applicable as  $f_{\&}(d(\neg b), d(\neg c))$ , and we can compute the degree of belief  $i(c)$  that we should increase  $d(c)$  as  $f_{\vee}(p_j, \dots, p_K)$ , i.e., as an aggregation for all the rules whose conclusion is this particular increase. Then, for every literal  $c$ , we have two conclusions: “update” with degree of belief  $i(c)$ , and “do not update” with the remaining degree of belief  $1 - i(c)$ .

If we interpret “update” as adding a constant  $\alpha$  to the previous degree of belief, then the standard defuzzification leads to the change from  $d(c)$  to the updated value  $d(c) + \alpha \cdot i(c)$ .

By applying this update procedure again and again, we will get an answer to a query.

**This method is successful and related to neural networks.** The resulting algorithm coincides with a heuristic method proposed by S. Maslov in [89], [90], [91] (see also [85], [87], [92], [86], [88]). Computer experiments has shown that this is indeed a very successful heuristic for expert systems.

This method was originally proposed based on the idea of simulating biological *neurons*, but it later turned out that exactly these same formulas follow from *fuzzy logic* heuristics, from the ideas of *chemical computing* (i.e., simulating chemical reactions), from heuristics of *numerical optimization* approach, from the ideas of *freedom of choice*, etc. (for a survey and latest results, see [86], [88]).

**3. CAN WE GUARANTEE THAT THE APPROXIMATING FUNCTION HAS THE DESIRED PROPERTIES (such as smoothness, simplicity, stability of the resulting control, etc.)?**

**For a model to be good, it must not only approximate the modeled system, but it must also preserve some properties of the modeled system.** In the previous sections, we have shown that an arbitrary system can be approximated by a model that originate from the fuzzy if-then rules. To be more precise, an arbitrary function  $u(x_1, \dots, x_n)$  that describes the dependence of the system’s output  $u$  on the inputs  $x_i$  can be approximated

by a function  $\tilde{u}(x_1, \dots, x_n)$  that originate from the fuzzy if-then rules. In these sections, to answer the question how good is this approximation, we simply compared the numerical values of the desired and actual output: if these two numerical values are  $\varepsilon$ -close (where  $\varepsilon$  is the desired accuracy), then we consider the approximation to be good.

If our only goal is to describe how the system reacts to different inputs, then predicting the system's reaction is all we need. In such applications, all we need is that the simulated output should be  $\varepsilon$ -close to the actual one. However, in many cases, the model is used to make *decisions*, e.g., to make *control* decisions. From this *control* viewpoint, a better way of comparing the approximate control and the desired control is not only by their values *per se*, but by the resulting behavior of the controlled system. It is therefore desirable to analyze whether we can always approximate a control strategy that has a certain property by a fuzzy control which has the same property. To answer this question, we will first enumerate the basic properties that we can require of the control, and then cite the corresponding universal approximation results.

Most of these results are applicable not only to control, but to the more general case of fuzzy rule based modeling as well.

**What properties do we require from control?** It is therefore desirable to see if fuzzy control can always approximate the desired control in this (more realistic) sense as well. To answer this question, let us first enumerate the basic properties that we can require of the control.

**First property: stability (control must control).** The main objective of the control is that it should control. For example, if we control a car on the road, then, for the largest part of the trip, one of the main objectives of this control is to make sure that it stays in its lane with the desired speed, i.e., that whenever it will accidentally deviate from the straight course, the steering control will return it back on course, and when the speed would deviate, the acceleration or deceleration would bring it back to the optimal cruise speed.

In the general case, we want a control that, after an initial deviation, will bring the controlled system back “on track”. This property is called *stability* of the control (and of the controlled system).

Stability is a matter of closed loop analysis: one and the same control strategy can be stable for one controlled system (described by a certain set of differential equations), but become unstable for a slightly different system.

Even for a fixed system, stability is a generic term. There are many different particular notions of stability, depending on how big initial deviations we allow (usually, only small ones), whether we want the system to

be stabilized for a potentially infinite amount of time or only for a given finite interval, etc.

**Second property: smoothness.** Stability is not all we expect from a control.

For example, when driving a car, stability means, in particular, that once the car swerved, it should return to the original trajectory. The faster it returns, the more stable is the system. Therefore, from the viewpoint of stability only, the ideal (optimal) control would be the one that brings the car back on track in the shortest possible time (i.e., with the largest possible acceleration). The resulting driving with sudden accelerations may be good on a racetrack or for a car chase, but it is very uncomfortable for passengers. From the passenger viewpoint, we prefer the resulting trajectory to be *smooth*.

The non-smoothness of the optimal control is not a peculiar feature of the car example: in control theory, there are general theorems that show that under certain (reasonably general) conditions, the optimal control is indeed of the above-described “bang-bang” type (see, e.g., [102]; not incidentally, the word “bang-bang” is an “official”, well-defined and widely used term in control theory).

Just like stability, smoothness is a generic notion; there are several different understandings (and formalizations) of what “smooth” means. In mathematical terms, *smoothness* is typically formalized as the existence of first, second, or higher order derivatives.

**Third property: computational simplicity.** Stability and smoothness are typical examples of the *idealized* goals. When, in mathematical control theory, we look for the optimal control strategy, we look for the optimal mathematical function, without taking into consideration how exactly we are going to implement this function.

In *real life*, however, the computational ability of the processor that actually computes the desired control is limited, so some very good control strategies may be too complicated for it. Moreover, in many control situations, we need the control *fast* (e.g., for a car control, if we spend too much time on the computation of the optimal control, the car may, by then, have already wrecked).

*In principle*, we can always replace the existing processor with a faster one, add extra memory, add an additional processor, etc., but this addition is *not* always *physically possible*:

- For example, in controlling a *space mission*, we are usually very much limited both in terms of the weight and the space required for a processor, and especially, in terms of the power feed; every increase in the

computational ability of the controlling processor can only come at the (undesirable) expense of the decrease in the useful payload.

- In commercial applications, e.g., in controlling an *appliance* (which one of the main areas of application for fuzzy control), one of the major considerations is *cost*. Every increase in the computational ability of the processor increases the cost of the simple appliance, and this cost increase is only reasonable if it leads to even better savings in performance<sup>4</sup>.

In view of all this, we would like our control to be *computationally simple*<sup>5</sup>.

**These properties are not exactly consistent: a trade-off is needed.**

At first glance, it may seem that we should require *all three* properties of our control. However, as we have mentioned, these properties are not exactly consistent:

- the most stable control is often not smooth at all;
- the most stable and the smoothest controls can be computationally complicated.

Therefore, in real life, we must either choose one of these properties that we desire the most, or, even better, look for some trade-off between these three properties.

**Can the approximating control preserve the desired property?**

We know that control strategies that result from applying fuzzy control methodology can be as close to the desired control as possible. For each of the three properties described above, we can ask whether the approximating control can have the desired property:

- If we are interested in the *stable* control, then it is natural to ask whether, for a given controlled system and for a given control  $u$  that stabilizes this system, the approximating control  $\tilde{u}$  can also be chosen to be stable (it is known in control theory that the very fact that  $\tilde{u}$  is close to the stable control  $u$  does not necessarily imply that  $\tilde{u}$  is stable as well).
- If we are interested in the *smooth* control, then it is natural to ask whether, for an arbitrary smooth function  $u$ , the approximating control  $\tilde{u}$  can also be chosen to be smooth. For smoothness, it is well known that a function that is close to the smooth one can be not smooth: e.g., the well-known Stone-Weierstrass theorem says that an arbitrary continuous function (not necessarily smooth one) can be, with an arbitrary

<sup>4</sup>We are greatly thankful to Piero Bonissone who attracted our attention to this example.

<sup>5</sup>Computational simplicity is what sometimes makes engineers use fuzzy control even in the situations when the system is well defined, and the optimal control is known.

accuracy, approximated by a polynomial (i.e., by a function that has derivatives of all orders and is thus maximally smooth).

- If we are interested in the *computationally simple* control, then it is natural to ask whether, for a given computationally simple function  $u$ , the approximating control  $\tilde{u}$  can also be chosen to be computationally simple (this question is not trivial, because it is easy to construct a very complicated function that is close to 0 or to any given simple one).

The answers to these three questions will be given in the following subsections.

### 3.1. IS FUZZY CONTROL A UNIVERSAL APPROXIMATION TOOL FOR STABLE CONTROLS?

Stability is one of the most important properties of a control strategy. Therefore, if we want to approximate a *stable* control strategy  $u(x_1, \dots, x_n)$ , we would like to be sure that the approximating strategy  $\bar{u}(x_1, \dots, x_n)$  is not only *close* to  $u$ , but that this approximating strategy  $\bar{u}$  is also *stable*.

The results described above only show that we can always find control rules for which the control strategy  $\bar{u}$  resulting from fuzzy control methodology is *close* to  $u$ , but this closeness does not automatically guarantee that the system controlled by  $\bar{u}$  is also stable.

In [111], [113], [114], [116], it is shown, for Takagi-Sugeno controllers, that, for each controlled system, for each control strategy which is stabilizing for this system, and for each  $\varepsilon > 0$ , we can find if-then rules for which the resulting fuzzy control strategy is  $\varepsilon$ -close to the given one and still stabilizing the given system. In these papers, stability is understood in some reasonable *practical* sense. For Mamdani-type control, the possibility of approximation by stable fuzzy controllers is proven in [113], [114], [119] (see also [112], [118]).

In [115], conditions are given under which the approximating control  $\bar{u}$  can be chosen as *stable* in a more theoretical sense (i.e, for an unlimited amount of time). In [117], a similar result is proven for *adaptive* fuzzy controllers.

### 3.2. IS FUZZY RULE BASED MODELING METHODOLOGY A UNIVERSAL APPROXIMATION TOOL FOR SMOOTH SYSTEMS?

**Smoothness is often important.** In many applications, we want the control to be *smooth*. Smoothness is important, e.g., in the following situations:

- In *aerospace engineering*, if we want to *dock* a spaceship to the space station, then, if we unnecessarily speed up, we'll crash into a space sta-

tion instead of smoothly approaching it. So here a reasonable criterion is *maximal smoothness*.

- For *public transportation*, we want a smooth control, because abrupt changes make the passengers feel very uncomfortable.
- For *robotic control*: For its movement, the robot is relying on its sensors whose outputs are used to constantly update the robot's image of the environment. For this update to be efficient, the robot must be able to identify obstacles and objects from his previous pictures on the new reading. If a robot moves smoothly, the change is gradual, and it is easier to trace the objects; if a robot make abrupt turns, then the new images are radically different from the old ones, and identification is much more difficult.

Smoothness is often useful not only in control applications, but in general modeling applications as well: if we know that a system is smooth, i.e., that the dependence on its output  $u$  on its inputs  $x_1, \dots, x_n$  is differentiable, then we want the model to be smooth too.

**Idea: how can we achieve smoothness?** According to the fuzzy rule based modeling methodology, the fuzzy model is obtained from the original membership functions (and, for Takagi-Sugeno approach, from the original models) by using:

- first, *&-operations*, to form the degree of firing of a rule;
- then, *aggregation* ( $\vee$ -)operations, to form the membership function for the output  $\mu_C(u)$ ;
- and, finally, *defuzzification*, to extract a single output value  $\bar{u}$  from the membership function  $\mu_C(u)$ .

In mathematical terms, the resulting fuzzy model is a composition of the corresponding functions and operations.

Therefore, to guarantee the smoothness of the resulting fuzzy model, all these functions and operations must be smooth: we must take smooth membership functions (and, in Takagi-Sugeno approach, smooth models), smooth  $\&$ - and  $\vee$ -operations, and a smooth defuzzification procedure.

**Splines are optimal.** The defuzzification procedure is usually smooth, so we only have to worry about the smoothness of the membership functions and  $\&$ - and  $\vee$ -operations. This problem was considered in [100], where it was shown, in particular, that the smoothest control (in some reasonable sense) or, in a more general case, the smoothest fuzzy model, are provided when the membership functions are *splines*, i.e., smooth piece-wise polynomial functions.

**How to make a smooth fuzzy model computationally simpler? (Mamdani approach).** Smoothness is only one of the desired properties of

the fuzzy model (and of fuzzy control), another is computational simplicity. So, it is desirable, within all possible methods that guarantee a certain level of smoothness, to find fuzzy rule based modeling methods that are computationally the simplest possible.

For that purpose, the authors of [109], [108] propose to use the so-called *B-splines* that have an additional advantages of being easy to compute.

If we choose  $f_{\&}(a, b) = a \cdot b$  as an &-operation, and, as input membership functions, we choose the splines  $\mu_{j,1}(x)$  that form the *partition of unity* (i.e., for which, for every  $x$ ,  $\sum_j \mu_{j,1}(x) = 1$ ), then the formula

$$\bar{u} = \frac{\sum r_j \cdot u_j}{\sum r_j},$$

where  $r_j = \mu_{j,1}(x)$ , turns into a *linear* formula  $\bar{u}(x) = \sum r_j \cdot u_j$  and thus, becomes *computationally easy*.

**Universal approximation results for these easy-to-compute smooth controls (Mamdani approach).** For B-splines, the possibility to approximate (and easily approximate) an arbitrary function  $f(x_1, \dots, x_n)$  by linear combinations of basic spline functions is well known; it is used, e.g., in computer graphics and computer-aided design (see, e.g., [95]).

**How to make a smooth fuzzy model computationally simpler? (Takagi-Sugeno approach).** To make the computations simpler, the authors of [104] use rules in which the right-hand side models are the simplest non-constant splines (i.e., linear functions), and the membership functions are also the simplest splines, i.e., piecewise-linear functions. To be more precise, they use Takagi-Sugeno model, with rules of the type “if  $x$  is  $A^j$ , then  $u = a_j \cdot x + b_j$ ” and triangular membership functions for fuzzy properties  $A^j(x)$ .

**Universal approximation results for these easy-to-compute smooth fuzzy models (Takagi-Sugeno approach).** In [104], it is proven that for Takagi-Sugeno model, with rules of the type “if  $x$  is  $A^j$  then  $u = a_j \cdot x + b_j$ ”, the resulting model  $\bar{u}(x) = (\sum (a_j \cdot x + b_j) \cdot \mu_{j,1}(x)) / (\sum \mu_{j,1}(x))$  with triangular membership functions  $\mu_{j,1}(x)$  can approximate an arbitrary twice differentiable SISO (single input single output) functions  $u(x)$  in the sense that not only  $\bar{u}(x)$  is close to  $u(x)$ , but the first and second derivatives  $\bar{u}'(x)$  and  $\bar{u}''(x)$  are close to the corresponding derivatives  $u'(x)$  and  $u''(x)$  of the approximated control.

If we only allow *homogeneous* Takagi model, with  $b_j = 0$  (i.e., if we allow only models that are typically used in traditional control), then we can only get universal approximation for the *first* derivatives [93].

**Smooth easy-to-compute fuzzy models has been successfully applied to real-life problems.** The resulting smooth fuzzy models have been successfully applied in control problems, e.g., in robotics [108], [110].

### 3.3. IS FUZZY RULE BASED MODELING A UNIVERSAL APPROXIMATION TOOL FOR COMPUTATIONALLY SIMPLE SYSTEMS?

**Computational simplicity is important.** In many real-life situations, especially in on-line control, it is important to compute the output as fast as possible.

**A fuzzy model is usually computationally simple, but sometimes, an even simpler model is needed.** The input-output relation produced by the fuzzy rule based modeling methodology is usually reasonably fast to compute (this is one of the major advantages of fuzzy rule based modeling), but in some situations, the resulting computation time is still too high (see, e.g., [107]). In such situations, it is desirable to approximate the function  $\bar{u}(x_1, \dots, x_n)$  resulting from the fuzzy rule based modeling methodology by a still simpler function  $\tilde{u}(x_1, \dots, x_n)$  that will be even easier to compute.

**Idea: second interpolation.** To make a fuzzy model computationally simpler, in [98], [94], [103], [101], [99], [96], it is suggested to use *piece-wise linear* or *piece-wise quadratic* functions  $\tilde{u}(x_1, \dots, x_n)$  to approximate the dependence between the inputs  $x_1, \dots, x_n$  and the output  $u$ . The author of [96] calls this approximation *second interpolation*.

In this case, the resulting fuzzy model is simply a piece-wise linear (or a piece-wise quadratic) function.

Piece-wise functions are a particular case of splines. Therefore, if piece-wise linear functions do not lead to the desired fuzzy model, we can consider splines (in particular, B-splines) of arbitrary order ([105], [106]).

**!!!Universal approximation results for computationally simple fuzzy models.** The question of universality of a computationally simple fuzzy rule based modeling can be reformulated as follows: *can we, with a given accuracy, approximate an arbitrary function  $u(x_1, \dots, x_n)$  by a piece-wise linear function?*

The possibility of such an approximation follows from known mathematical results (for details, see [96], [97]; since piece-wise functions are a particular case of splines, this result is a particular case of the general result of approximability by splines.)

## 4. AUXILIARY APPROXIMATION RESULTS

### 4.1. AN OVERVIEW

In the above description of fuzzy rule based modeling methodology, we assumed the following:

- first, that the only expert knowledge that we can use consists of expert rules;
- second, that these rules have the simple “if-then” form (“if ⟨something⟩ and ⟨something⟩, then ⟨something⟩”);
- and finally, that we are only interested in the *result* of the tuning, but not in the very *process* of tuning.

In real life, of course, these assumptions may turn out to be false:

- first, in addition to expert rules, we may have additional expert information about the modeled system;
- second, the expert information can be formulated in a more complicated way than simply “if-then” rules;
- finally, we are definitely interested not only in the result of the tuning, but also in the process of tuning.

For these situations, it is also possible to formulate and prove universal approximation results. These results will be described in the following sections.

### 4.2. USING ADDITIONAL EXPERT INFORMATION (FUZZY MODELS AND FUZZY CONTROL RULES)

In the above text, we showed that the fuzzy rule based modeling methodology has the universal approximation property. The main applications of fuzzy rule based modeling methodology can be divided into two groups:

- designing the *model* of a system, i.e., a function  $y(x_1, \dots, x_n)$  that simulates the way the modeled system functions;
- designing the *control strategy* for a given system, i.e., a function  $u(x_1, \dots, x_n)$  which simulates the way an expert controller controls the system.

In the applications from the first group, the fuzzy rule based modeling methodology transforms the if-then rules that describe how the inputs affect the output of the system, into a system’s model. In fuzzy control applications, this methodology transforms the if-then control rules into a control strategy.

In some control situations, however, in addition (or instead of) fuzzy control if-then rules, we have if-then rules that constitute a fuzzy model of the controlled system. For example, an expert in driving, in addition to

the rules that say how to best control a car, can also predict how the car will react to different (not necessarily best) controls; this prediction will be formulated in terms of rules like “if you break hard on a slippery road, the car will most probably swerve”. How can we use this additional information (i.e., fuzzy model) in designing fuzzy control?

In [123], the universal approximation theorem is used to show that, *in principle*, it is possible to formulate a “universal fuzzy controller”, i.e., a device that, given the description of the system and the control objective, describes the optimal control strategy. This proof, however, does not provide us with a feasible algorithm for actually *constructing* such a system. Moreover, as shown in [120], the problem of finding the optimal control strategy is, in general, computationally intractable (NP-hard). Feasible constructions are known for simple fuzzy models ([124], [122]); it is desirable to extend these constructions to more complicated (and thus, more realistic) fuzzy models [121].

#### 4.3. EXPERT RULES THAT USE UNUSUAL LOGICAL CONNECTIVES

Fuzzy rule based modeling methodology translates the expert’s knowledge about the system, that is formulated in “fuzzy” terms of natural language, into a precise model. In many real-life situations, this knowledge is already formulated in terms of if-then fuzzy rules; in fuzzy rule-based modeling methodology, these rules are then, usually, transformed into statements that only use connectives “and”, “or”, and “not”.

However, in many other real-life cases, the fuzzy knowledge about the system can be of much more general type than if-then rules. In these cases, a question appears: can we approximate an arbitrary fuzzy knowledge, that uses *arbitrary* logical *connectives* (including, e.g., different versions of fuzzy implication), by a knowledge described in terms of “and”, “or”, and “not” fuzzy connectives? In other word, *can we approximate an arbitrary fuzzy logical connective by a combination of these three basic ones?*

It may seem, at first glance, that different unusual connectives are purely mathematical constructions, but, as shown, e.g., in [128], even those connectives that may appear this way actually result from very natural axioms. In view of this result, it is desirable to consider the approximability of *arbitrary* logical connectives.

It turned out ([127], [126]) that in general, such an approximation of an arbitrary connective *is* possible, but only when we, in addition to these three basic connectives, allow *modifiers* such as “very”, “slightly”, etc. (that without the modifiers, such an approximation is impossible, is also shown in [125]).

## 4.4. TAKING THE LEARNING PROCESS INTO CONSIDERATION (FUZZY NEURAL NETWORKS)

**How can we tune?** Traditional universal approximation results only show that *in principle*, we *can* always approximate any given system by a fuzzy model that is obtained by applying fuzzy rule based modeling methodology to appropriate if-then rules. These results does not show *how* to find these rules, i.e., how to tune the original model into the desired one. So, in addition to the universal approximation results, it is desirable to show the universality of some of these *tuning methods*.

Where can these universal tuning methods come from? To answer this question, let us recall that we are talking about tuning methods for *fuzzy rule based modeling*. By definition, fuzzy control is a methodology for *simulating* the expert's knowledge in the computer. Therefore, in order to tune the fuzzy control, it is natural to simulate the way we humans "tune" our knowledge.

**Neural networks: simulating human brain.** At first glance, it may seem that we can use the same fuzzy rule based modeling methodology to tuning as well: just ask the experts how they tune their rules, and then use this methodology to translate these rules into an actual tuning strategy. The main problem with this idea is that fine-tuning of the knowledge is even harder to explain in words than the knowledge itself. This fine-tuning is done mainly on an *intuitive* level.

Therefore, in order to simulate how this fine-tuning is done, we cannot rely on the experts' statements, we have to actually simulate how this fine-tuning is done in the human brain.

The main "processing units" of human brain are neurons. Therefore, in order to simulate this "brain processing", we must simulate the neurons. Each biological neuron has several inputs (up to  $10^4$ ), and a single output. Neural processing starts with sensors sending signals to the neurons. The intensity of the original signal is represented by the frequency of the pulses generated by the sensor. Signals  $x_1, \dots, x_n$  coming from different inputs are "weighted" differently, dependent on the thickness of the connection. As a result, the total number of pulses per time unit that get into the neuron is approximately equal to the weighted sum  $x = w_1 \cdot x_1 + \dots + w_n \cdot x_n$ , where  $w_1, \dots, w_n$  are weights of different inputs. The output of a neuron depends on its *threshold* level  $w_0$ :

- if the total weighted number of input pulses  $x$  is much larger than  $w_0$ , the neuron gets maximally excited and emits the largest possible number of pulses;
- if the total weighted number of input pulses  $x$  is much smaller than the threshold level  $w_0$ , the neuron does not get excited at all, and does

not emit any pulses at all;

- if the total weighted number of input pulses  $x$  is approximately equal to the threshold value  $w_0$ , the neuron gets excited only partly.

Crudely speaking, the output signal  $y$  depends on the difference  $z = x - w_0$  between the weighted sum  $x$  of inputs and the threshold value  $w_0$ :  $y = s_0(x - w_0)$ ; if the difference  $z$  is  $\gg 0$ , then  $s_0(z) \approx 1$ ; if  $z \ll 0$ , then  $s_0(z) \approx 0$ , and in general,  $s_0(z)$  monotonically increases. So,  $y = s_0(z) = s_0(x - w_0) = s_0(w_1 \cdot x_1 + \dots + w_n \cdot x_n - w_0)$ .

Experimentally, the input-output relation for biological neurons is best represented by the so-called *logistic* function  $s_0(z) = 1/(1 + \exp(-z))$ . (This choice turns out to be optimal in some reasonable sense, see, e.g., [142], [144]).

Signals coming from the neuron enter other neurons, etc., until we get the final processing results. Artificial (simulated) neurons try to simulate the same structure. In the most widely spread artificial neurons, the output signal  $y$  is connected with the input signals  $x_1, \dots, x_n$  by the same formula  $y = s_0(w_1 \cdot x_1 + \dots + w_n \cdot x_n - w_0)$ .

It is known that neural networks are *universal approximation* tools (see, e.g., [135], [131], [132], [133], [136], [130], [129], [137], [138], [141], [145], [150], [139], [140], [146], [147], [133], [143], [148], [149], [151], [134]), i.e., that an arbitrary continuous function can be, with an arbitrary accuracy, approximated by a function computed by a neural network<sup>6</sup>.

**Fuzzy neural networks.** Neural networks input well-defined (numerical) inputs and return numerical outputs. They are therefore good in approximating (crisp) functions  $f(a_1, \dots, a_n)$  from real numbers to real numbers. In other words, if we have crisp if-then rules of the type “if  $x_1 = a_1^{(1)}, \dots, x_n = a_n^{(1)}$  then  $u = f(a_1^{(1)}, \dots, a_n^{(1)})$ ”, “if  $x_1 = a_1^{(2)}, \dots, x_n = a_n^{(2)}$  then  $u = f(a_1^{(2)}, \dots, a_n^{(2)})$ ”, ..., then we can use a standard neural network.

Fuzzy rule based modeling is dealing with the situations in which both conditions and conclusions of the if-then rules are not *crisp* numbers, but *fuzzy* sets. Thus, to tune fuzzy rules, we must modify neural networks so that they will be able to process fuzzy sets instead of numbers.

<sup>6</sup>The paper [152] raises an important concern: Most of the universal approximation results do not take into consideration the precision with which computers represent and process real numbers. The authors of this paper show that in order to get an approximation of arbitrary accuracy, we must have operations with real numbers performed with a much larger precision than in the standard computers. There is, therefore, a need to develop more realistic universal approximation results in which the computer precision is also taken into consideration. Such results are given, e.g., in [149].

A mathematician reader should be warned that the paper [152] is written by non-mathematicians, and is, therefore, somewhat terminologically confusing: e.g., when talking about *polynomials*, the authors of this paper also mean infinite Taylor series, etc.

This can be easily done if we allow fuzzy sets in the main formula that defines a neuron, i.e., if we consider a neuron as a device that transforms fuzzy inputs  $X_1, \dots, X_n$  into the fuzzy output

$$Y = s_0(W_1 \cdot X_1 + \dots + W_n \cdot X_n - W_0),$$

For the resulting *fuzzy neural networks*, we can formulate a similar question: are they universal approximation tools? Can they approximate arbitrary functions from fuzzy sets to fuzzy sets?

This question was first analyzed in [153], [155], with a somewhat unexpected result: fuzzy neural networks are *not* universal approximations tools for functions from fuzzy sets to fuzzy sets. The reason for this negative result is rather intuitively clear: each neuron of the above type (and thus, each neural network formed by such neurons) is *monotonic* in the sense that if we “widen” the inputs, the output fuzzy sets grows “wider”, while an arbitrary function from fuzzy sets to fuzzy sets need not necessarily be monotonic. For certain classes of monotonic functions (e.g., for all functions generated by number-to-number ones) fuzzy neural networks are indeed universal approximation tools [155].

To get approximation property in the most general case, it is necessary to replace, in the formulas for a fuzzy neuron, standard fuzzy arithmetic by a more general one; with this replacement, the universal approximation property becomes true [154], [156].

A detailed description of fuzzy neural networks is given in Chapters 8 (“Learning and tuning”), 9 (“Neuro fuzzy systems”), and 10 (“Neural networks and fuzzy logic”).

## 5. HOW TO MAKE THE APPROXIMATION RESULTS MORE REALISTIC

### 5.1. A GENERAL OVERVIEW

**The existing universal approximation results are of theoretical nature.** The universal approximation results that we have described are of great fundamental importance. These results show that, in principle, for every function  $u(x_1, \dots, x_n)$ , and for every accuracy  $\varepsilon > 0$ , there exists a set of rules for which the application of the fuzzy rule based modeling methodology leads to an output  $\bar{u}(x_1, \dots, x_n)$  that is  $\varepsilon$ -close to  $u$ .

**How can we make these results more practical?** It is desirable to use the theoretical universal approximation results to actually construct the fuzzy models and use them, e.g., in real-life control. This is, however, not always possible.

Since the universal approximation theorems were mainly motivated by *fundamental, methodological* questions, the authors of these theorems were

mainly interested in showing that *some* rules existed, and not so much in looking for *realistic* sets of rules. As a result, the sets of rules presented by these theorems are not always very realistic:

- First, these results often *solve a somewhat oversimplified problem*. To be more precise, the main universal approximation results are based on the assumption that we start with the *exact* values of the input variables  $x_1, \dots, x_n$ . In reality, sensors are never 100% accurate. In practical applications, this inaccuracy must be taken into consideration.
- Second, *the solution* provided by the major universal approximation results *is* often *too complicated*. To be more precise, there are often unrealistically many rules in the rule base. This abundance of rules makes the resulting fuzzy model (in particular, fuzzy control) computationally non-realistic.

In the following two sections, we will show how we can solve these two problems and thus, how to make the universal approximation results more realistic (and hence, more practically useful).

## 5.2. INACCURACIES IN THE INPUT DATA

**Two possible types of inaccuracy: fuzzy and interval.** Traditional fuzzy rule based modeling methodology is based on the assumption that we know the *exact* values of the input quantities  $x_1, \dots, x_n$ . In real life, these values either come from measurements, or from expert estimates. In both cases, the values are not 100% precise:

- *Measurements* are never 100% accurate, there always is a possibility of a measurement inaccuracy; the best measurements are the ones for which this inaccuracy is the smallest possible. The manufacturer of the measuring instrument usually supplies it with a *guaranteed* accuracy, i.e., with an upper bound  $\Delta$  on the possible error values. As a result, if, as a result of measuring a certain quantity  $x$ , we get the value  $\tilde{x}$ , the only thing that we can now conclude about the actual value  $x$  is that this actual value differs from  $\tilde{x}$  by at most  $\Delta$ , i.e., that  $x$  belongs to the interval  $[\tilde{x} - \Delta, \tilde{x} + \Delta]$ . In other words, instead of the precise value of  $x_i$ , we have a (crisp) *set*  $X_i$  of possible values of  $x_i$ .
- If the values  $x_i$  come from *expert estimates*, then these expert estimates also never absolutely accurate. At best, an expert may estimate the actual value of the quantity by saying something like “it is approximately equal to 1”. Since we are talking about fuzzy rule based modeling, in fuzzy modeling, such statements are naturally formalized in terms of fuzzy sets. Thus, instead of a precise value  $x_i$ , we get a (fuzzy) set  $X_i$  of possible values of  $x_i$ .

In both cases, we have a set  $X_i$  (in general, a fuzzy set) of possible values of  $x_i$ . Based on these sets, we must select the output values.

**Fuzzy inaccuracy.** In case all inputs were supposed to be precisely known, the desired fuzzy model was a function that maps each tuple of real numbers  $(x_1, \dots, x_n)$  (i.e., possible values of the inputs) into a real number (corresponding output).

If we take into consideration that inputs have fuzzy uncertainty, then, in mathematical terms, we need, instead of a mapping from tuples of real numbers into real numbers, we need a mapping that maps each tuple of *fuzzy sets*  $(X_1, \dots, X_n)$  (representing the input) into a real number (the output value). (If we are interested not in the automated control, but in the recommendations for the human controller, then it is sufficient to produce a fuzzy set.)

A methodology that uses fuzzy rules to generate such mappings is proposed and used in [157], [158], [162], [163], [164], [165], [166], [167].

**Interval inaccuracy.** An important particular case of fuzzy inaccuracy is *interval* inaccuracy. This type of uncertainty occurs if we take measurement inaccuracy into consideration. If we take it into consideration, then the resulting fuzzy model becomes more smooth and the resulting fuzzy control becomes more stable and more smooth (see, e.g., [159], [171], [161], [172], and references therein). Let us illustrate this phenomenon on the example of fuzzy control.

- Traditional fuzzy control techniques, if used appropriately, lead to a control that is stable. However, if we use only a *single* value  $\tilde{x}_i$  from the interval  $[\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$  of possible input values, we will get a control that is *stable for this particular value*, and may *not* be *stable* at all for the (unknown) *actual* value. The only way to guarantee that the control is stable for the *actual* (unknown) value is to guarantee that it is stable for *all* values from this interval. This requires, at least, that the algorithm that computes the control values should have this interval at its disposal.
- A measured value  $\tilde{x}_i$  is, in general, unpredictably (“randomly”) different from the actual value  $x_i$ . As a result, the control  $\bar{u}$  based on the measured value will “wobble” around the control that correspond to the actual (unknown)  $x_i$ . The random wobbling around a smooth process usually makes it less smooth. Thus, the way to avoid this wobbling (and to make control smoother) is to take into consideration that the actual values are within the *intervals*, and then, to choose the *smoothest* possible control within these intervals.

**Is the set-valued fuzzy rule based modeling methodology a universal approximation tool?** It is interesting to check whether this set-

valued fuzzy control methodology is still a universal approximation tool, i.e., whether an arbitrary function from tuples of fuzzy sets into real numbers (or into fuzzy sets) can be approximated by functions stemming from some rule bases.

The universal approximation property was first proven in [158] for the following special type of fuzzy-input situation: When rules are of the type “if  $x_1$  is  $A_1^j, \dots$ , and  $x_n$  is  $A_n^j$ , then  $B^j(u)$ ”, and the inputs are not known exactly, but represented instead by fuzzy sets  $X_1, \dots, X_n$ , then the output  $\mu_C(u)$  is defined as a fuzzy set  $(B^{j_1}(u) + \dots + B^{j_t}(u))/t$ , where  $j_k, 1 \leq k \leq t$ , are all the rules  $j$  for which  $d(X_i, A_i^j) \geq \alpha$  for some “set distance”  $d$  and for some threshold  $\alpha > 0$ .

A general result, that “non-singleton” (i.e., set-valued) systems can approximate an arbitrary continuous function from compact sets to real numbers with an arbitrary accuracy, was proven in [167].

The universal approximation property for fuzzy inputs is somewhat clarified by results presented in [168], [160], [169], [170]. Namely, in these papers, it is shown that there are functions from fuzzy sets to fuzzy sets (and even from intervals to intervals) that *cannot* be approximated not only by compositions of membership functions and “and” and “or” operations, but also by *any* compositions of functions of one and two variables. However, if we add functions of three or more variables, e.g., the “dot” product  $(\sum u \cdot \mu_C(u) \text{ or } \sum r_j \cdot u_j)$  of the type used in centroid defuzzification, then we can approximate arbitrary smooth functions with a reasonable accuracy.

### 5.3. CAN WE HAVE FEWER RULES?

**Too many rules.** We have already mentioned that the fuzzy if-then rule base provided by the major universal approximation results is often too complicated and contains unrealistically many rules. Indeed, to make the model better, we must take into consideration all inputs.

The more inputs we have, the more rules we need to describe: if we have  $v$  variables and we only consider 3 possible values of each variable (e.g., negative, negligible, and positive; or small, medium, and large), then we must consider  $3^v$  rules that describe the output for all  $3^v$  combinations. Even for a toy robot, the number of inputs  $v$  can be in the dozens. In this case,  $3^v$  becomes unrealistically large.

So, in theory, fuzzy rule based modeling is possible, but in practice, we are often still far from the actual model design or controller design (see, e.g., [180], [181], [182], [185], [179], [184], [197]).

The estimates for an *accuracy* of an approximation with a given number of rules are given also in [194], [195], [196].

**In general, we cannot approximate by using fewer rules.** One consolation is that this abundance of the rules is not a drawback of fuzzy rule based modeling: it is caused simply by the fact that there are too many functions. Whether we consider arbitrary continuous functions, or smooth (differentiable) functions with a bound on the derivative, or any other reasonable class of functions that we want to approximate, there are simply too many functions in each class.

In precise mathematical terms, if we want to approximate these functions with an accuracy  $\varepsilon > 0$ , and we are given two functions  $f \neq f'$  whose distance is greater than  $2\varepsilon$ :  $d(f, f') > 2\varepsilon$ , then these two functions cannot be approximated by the same functions  $F$ : otherwise, we would have  $d(f, F) \leq \varepsilon$ ,  $d(f', F) \leq \varepsilon$ , and  $d(f, f') \leq d(f, F) + d(F, f') \leq 2\varepsilon$ . Hence, we need two different functions to approximate  $f$  and  $f'$  with the desired accuracy.

Similarly, if we have  $N$  different functions such that every two of them are more than  $2\varepsilon$  apart, we will need  $N$  different approximating functions. How many bits do we need to represent this information? If we use  $k$  bits, then by using all  $2^k$  combinations of  $k$  0's and 1's, we can represent  $2^k$  different objects. Thus, to represent  $N$  different objects, we need at least  $\log_2(N)$  bits. For a given class  $\mathcal{F}$  of functions, the value of this logarithm for the largest possible  $N$  is called an  $\varepsilon$ -*capacity* of this class ([191], [186]). By explicitly constructing the sets of functions with large  $N$ , researchers have shown that the  $\varepsilon$ -capacity of different classes of functions  $f(x_1, \dots, x_n)$  of  $n$  variables grows at least exponentially (i.e., as  $c^n$ ) with the number of inputs  $n$  (see, e.g., Chapter 10 of [191] or [175]).

Thus, in general, there is no way to approximate all the functions from one of these classes without having to spend an unrealistic (exponential) amount of memory just to store the approximating functions.

**There is hope.** At first glance, the impossibility to approximate an arbitrary function by an appropriate fuzzy rule based model with a realistic number of rules sounds like a negative result. But in many real-life situations, we *do* have successful fuzzy models and successful fuzzy controllers. It is, therefore, desirable to formulate a reasonable class of function for which an approximating fuzzy model with a few rules *is* possible.

An approximation with fewer rules was proposed, for 1-D case, in [189], [190], and for a general case, in [174]: Whereas in the previous approximations, the number of rules increases when we need a better accuracy, in this new approximation scheme, the number of rules remains fixed as the accuracy increases, and in principle, we can get a precise description of the original input-output function, not just an approximation. Moreover, if we allow non-convex membership functions, then we can approximate an arbitrary function by using only two rules ([177], [192], [174], [193], [184]). The

downside of these results is that while the standard approximation results use, say, triangular membership functions that are easy to store and to process, this new scheme uses special membership functions that, in essence, code the desired input-output function. So, while we get fewer rules, we do not automatically decrease the total amount of information that needs to be stored.

An idea of actually decreasing this amount of information is described in [180], [181], [182], [185], [176], [184]: it consists of building the rules around the extrema of the input-output function. In these papers, it is shown that if we fix the number of rules, then (under a certain reasonable criterion) locating the membership functions around the extrema indeed leads to the best approximation. Clustering, neural learning, and other learning techniques can then be used to fine-tune the resulting rule bases. An (empirically supported) conjecture is that if an input-output function has few extrema, we will be able to approximate this function with much fewer rules than in the general case.

In [183], another idea is proposed: The expert if-then rules, with which the traditional fuzzy rule based modeling methodology starts, determine the control  $u$  based on the *current* values of the inputs  $x_1, \dots, x_n$ . In real life, when experts predict the output of a system, they use not only the information about the current state of the modeled system (i.e., about the inputs  $x_1, \dots, x_n$ ), but also about the *previous* output: e.g., a rule can be “if the inputs did not change much, the output will be almost the same as before”. The corresponding expert if-then rules may, therefore, include not only conditions on the current inputs  $x_i(t)$ , but also conditions on the previous value of the output  $u(t-1)$ . If we apply the standard fuzzy rule based modeling methodology to these rules, we get an input-output relation of the type  $\bar{u}(t) = \bar{u}(x_1(t), \dots, x_n(t), u(t-1))$ . The possibility of such a *feedback* can, often, drastically decrease the number of rules required for an approximation with a given accuracy. The main *drawback* of feedback systems is that the resulting controls are often unstable, so a special care needs to be taken to guarantee the system’s stability.

That the hope (of finding reasonable classes of functions for which fewer rules are sufficient) is realistic can be shown on the example of another approximation scheme: neural networks. The negative result formulated above is not only about fuzzy rule based modeling approximations, it is applicable to other means of approximation such as neural networks, wavelet, etc. For neural network, this negative result shows that for some functions, we need exponentially many neurons to approximate. However, in practice, often reasonably few neurons are sufficient for a good approximation. Recently, this empirical fact has been theoretically explained: a reasonable class of functions has been described for which the necessary number of neurons

remains quite feasible [178], [173], [187], [188].

## 6. FROM ALL FUZZY RULE BASED MODELING METHODOLOGIES THAT ARE UNIVERSAL APPROXIMATION TOOLS, WHICH METHODOLOGY SHOULD WE CHOOSE?

**At first, there was hope that universal approximation results can help to choose a fuzzy rule based modeling methodology.** We have already mentioned that the universal approximation property is an important feature of fuzzy rule based modeling methodology. In view of this importance, when several different variants of this methodology were proposed, variants that use different membership functions, different “and” and “or” operations, and different defuzzification procedures, the hope was that only few of these variants will have the universal approximation property, and that this property will thus help us to select the appropriate methodology.

For example, when Bart Kosko proved his first universal approximation result, that a methodology based on  $+$  as aggregation (“or”) operation is a universal approximation tool, and proved in a manner that essentially used the properties of  $+$ , he naturally conjectured that this result may show the advantage of  $+$  over other aggregation rules (such as  $\max$ ).

**It turned out that the universal approximation property itself is not a good choice criterion; so, how can we choose?** As we have seen in the previous sections, most variants of the fuzzy rule based modeling methodology have the universal approximation property. Thus, the very fact that a variant possesses this property does not make it much better than many other variants.

This does not mean, of course, that some variants are not better than the others. Indeed, as we have mentioned, a fuzzy model is not always perfect:

- it can have too many rules, making it computationally non-realistic; or
- the resulting model can be not of very good quality (depending on what we want from the model, it can mean not very smooth, or, in control applications, not very stable, etc.)

So, out of all variants of fuzzy rule based modeling methodology that have the universal approximation properties, it is natural to select the *best* variant, i.e., the variant for which:

- either the *smallest* number of rules is needed, on average, to approximate the given control with a given accuracy; or,

- the resulting model is *the best* according to the chosen criterion (i.e., is the most *smooth*, or, in case of control, the resulting control is the most *stable*, etc.).

In this section, we will briefly describe the situations in which the best choice is known.

#### 6.1. MAIN RESULTS. PART I. BEST APPROXIMATION

**Choice of membership functions.** The authors of [206], [207], compared the quality of the approximation achieved by using different shapes of membership functions. Their numerical experiments have shown that in almost all test situations, the best approximation if we use the “sinc” membership function  $\sin(x)/x$ .

The paper [199], contains a partial explanation of this result: namely, it is proven that in linear approximation, the function  $\sin(x)/x$  is indeed the best (in some reasonable sense). It is desirable to extend this explanation to the general (non-linear) case.

**Choice of “and” and “or” operations.** In [218], is is shown that the choice of the product  $a \cdot b$  as an “and” operation leads to a better approximation than the choice of the minimum  $\min(a, b)$ .

**Choice of defuzzification.** In [218], is is shown that the choice of the centroid defuzzification leads to a better approximation than the *Mean of Maximum* defuzzification.

#### 6.2. MAIN RESULTS. PART II. BEST MODEL

**Choice of membership functions.** The most *robust* membership functions (i.e., the least sensitive to the inaccuracy of the input data) are piecewise-linear ones [208], [210].

This result explains why the piecewise-linear membership functions are, at present, most frequently used.

**Choice of “and” and “or” operations.** (These results are (mainly) summarized in [203], [204], [208], [210], [217], [198].)

- If we are looking for the *smoothest* model, then the best choice is to use  $f_{\&}(a, b) = a \cdot b$  and  $f_{\vee}(a, b) = \min(a, b)$  [203], [204], [217].
- If we are looking for the model that is *most robust* (i.e., least sensitive to the inaccuracy with which we measure the membership functions), then, depending on what exactly we are looking for, we can get two different results:

- if we are looking for the model that is the most robust *in the worst case*, then the best choice is to use  $f_{\&}(a, b) = \min(a, b)$  and  $f_{\vee}(a, b) = \max(a, b)$  [209], [211], [208], [210], [213];
- if we are looking for the model that is the most robust *in the average*, then the best choice is to use  $f_{\&}(a, b) = a \cdot b$  and  $f_{\vee}(a, b) = a + b - a \cdot b$  [212], [208], [210], [213];
- instead of minimizing the *average* error, we can try to minimize the corresponding *entropy* [214], [215], [216], [201], [202], [200]:
  - \* if we use the *average* entropy (in some reasonable sense), we get the same pair of optimal functions as for average error;
  - \* for an appropriately defined *worst-case* entropy the optimal operations are  $f_{\&}(a, b) = \min(a, b)$  and  $f_{\vee}(a, b) = a + b - a \cdot b$ .
- If we are looking for the model that is the *fastest to compute*, then the best choice is to use  $f_{\&}(a, b) = \min(a, b)$  and  $f_{\vee}(a, b) = \max(a, b)$  [205].
- Finally, if, in control applications, we are looking for the *most stable* control for a given system, then the best choice is to use  $f_{\&}(a, b) = \min(a, b)$  and  $f_{\vee}(a, b) = a + b - a \cdot b$  [203], [204], [217], [202].

**Choice of defuzzification.** In [203], [204], [202], [200], we show that the optimal defuzzification is given by the centroid formula.

*Comment.* These optimization results are in good accordance with the general group-theoretic approach that enables us to classify techniques that are optimal relative to arbitrary reasonable criteria [203], [204], [217], [198].

## 7. A NATURAL NEXT QUESTION: WHEN SHOULD WE CHOOSE FUZZY RULE BASED MODELING IN THE FIRST PLACE? AND WHEN IS, SAY, NEURAL MODELING BETTER?

In the previous section, we mentioned that initially, there was hope that only a few variants of the fuzzy rule based modeling methodology would have a universal approximation property and that therefore, this property would help us select the right variants. Alas, it turns out that most of the variants are universal approximation tools, and thus, we still face the problem of how to choose the best variant. In the previous section, we showed how this problem (of choosing the best fuzzy rule based modeling methodology) can be solved.

Within the scope of this book, it is natural to restrict ourselves to the fuzzy rule based modeling methodology. However, from the practical viewpoint, there is no reason why we should restrict ourselves to this methodology only and not consider other intelligent modeling methodologies that

are also known to be universal approximation tools. If we do consider these other methodologies, then we face a similar question: *which of the intelligent modeling methodologies should we choose?* Fuzzy? neural? or some new (yet to be developed) methodology?

There are two ways to compare these two intelligent modeling methodologies:

- First, we can compare the *time* and effort that it takes *to generate* the output of a model.
- Second, we can compare the quality of the resulting model.

For the first way, the answer is easy: neural networks, usually, require lots of time to tune, while fuzzy rule based methodology immediately leads to a reasonable model. So, from the viewpoint of time and effort necessary to generate a model, fuzzy rule based modeling wins hands down.

If, however, we want to compare the *quality* of the resulting models, then the comparison is not so straightforward. The answer depends on which of three quality criteria we use for comparison: smoothness, computational simplicity, or (for control applications) stability. Let us analyze this problem for these three criteria.

### 7.1. SMOOTHNESS AND STABILITY

Smoothness is the easiest to analyze:

- For each *neuron*, the dependency on the output  $y$  on the inputs  $x_1, \dots, x_n$  described by the formula  $y = s_0(w_1 \cdot x_1 + \dots + w_n \cdot x_n - w_0)$  with a smooth function  $s_0(z) = 1/(1 + \exp(-z))$ . Thus, the dependence between the inputs and the output is smooth. The transformation from the sensor inputs  $x_1, \dots, x_n$  into the output value  $u$  that is performed by a neural network is a composition of (smooth) transformations performed by individual neurons, and is, therefore, smooth. Hence, the model generated by a neural network is always smooth.
- The model generated by *fuzzy rule based methodology* is a composition of the membership functions, “and” and “or” operations, and of the defuzzification procedure. The membership functions used in fuzzy rule based modeling are often non-smooth (e.g., triangular); often, non-smooth “and” and “or” operations are also used, such as min for “and” and max for “or”. As a result, the model  $\tilde{u}(x_1, \dots, u_n)$  generated by the fuzzy rule based methodology is often non-smooth.

We have shown (in the above text) how we can gain *some* smoothness by using smooth membership functions and smooth “and” and “or” operations, but typically, the resulting model is *not* infinitely differentiable, and thus, the neural network model is smoother.

Thus, we can conclude that the neural network modeling leads to a smoother model.

This analysis of smoothness can help us to analyze stability of control applications. We have already mentioned that, crudely speaking, stability and smoothness are opposites: the maximally stable control is the least smooth, and vice versa. If we take into consideration that the more stable the control, the less smooth it is, we can then also conclude that the neural network control is the least stable one.

## 7.2. COMPUTATIONAL COMPLEXITY

**Often, a model must allow fast output computations.** One of the main objectives of modeling is to predict how a modeled system will evolve and how it will react to different changes in the environment. In many real-life applications of modeling (e.g., in many control applications) the system evolves pretty fast, so we need the modeling results fast. In other words, we must be able, given the values  $x_1, \dots, x_n$  of the input variables, to compute the system's simulated output  $u$  in the shortest possible time.

The model's output depends on the values of the system's inputs. So, to get a high quality model, we must take into consideration as many parameters that affect the actual system's output as possible. The more inputs we take into consideration, the more numbers we have to process, so, the more computation steps we must perform. So, in many real-life problems, high-quality modeling is a real-time computation problem with a serious time pressure.

**Parallel computing is an answer.** A natural way to increase the speed of the computations is to perform computations *in parallel* on several processors. To make the computations really fast, we must divide the algorithm into parallelizable steps, each of which requires a small amount of time.

What are these steps?

**The fewer variables, the faster.** As we have already mentioned, the main reason why modeling algorithms are computationally complicated is that we must process many inputs. For example, modeling or controlling a car is easier than controlling a plane, because the plane (as a 3-D object) has more characteristics to take care of, more characteristics to measure and hence, more characteristics to process. Modeling or controlling a space shuttle, especially during the lift-off and landing, is even a more complicated task, usually performed by several groups of people who control the trajectory, temperature, rotation, etc. In short, the more numbers we need to process, the more complicated the algorithm. Therefore, if we want to decompose our algorithm into fastest possible modules, we must make each module to process as few numbers as possible.

**Functions of one variable are not sufficient.** Ideally, we should only use the modules that compute functions of one variable. However, if we only have functions of one variables (i.e., procedures with one input and one output), then, no matter how we combine them, we will always end up with functions of one variable. Since our ultimate goal is to compute the modeling function  $u = f(x_1, \dots, x_n)$  that depends on many variables  $x_1, \dots, x_n$ , we must therefore enable our processors to compute at least one function of two variables.

What functions of two variables should we choose?

**Choosing functions of two variables.** Inside the computer, each function is represented as a sequence of hardware implemented operations. The fastest functions are those that are computed by a single hardware operation. The basic hardware supported operations are: arithmetic operations  $a+b$ ,  $a-b$ ,  $a \cdot b$ ,  $a/b$ , and  $\min(a, b)$  and  $\max(a, b)$ . The time required for each operation, crudely speaking, corresponds to the number of bits operations that have to be performed:

- Division is done by successive multiplication, comparison and subtraction (basically, in the same way as we do it manually), so, it is a much slower operation than  $-$ .
- Multiplication is implemented as a sequence of additions (again, basically in the same manner as we do it manually), so it is much slower than  $+$ .
- $-$  and  $+$  are usually implemented in the same way. To add two  $n$ -bit binary numbers, we need  $n$  bit additions, and also potentially,  $n$  bit additions for carries. Totally, we need about  $2n$  bit operations.
- Computing the minimum  $\min(a, b)$  of two  $n$ -bit binary numbers  $a$  and  $b$  can be done in  $n$  binary operations: we compare the bits from the highest to the lowest, and as soon as they differ, the number that has 0 as opposed to 1 is the desired minimum: e.g., the minimum of 0.10101 and 0.10011 is 0.10011, because in the third bit, this number has 0 as opposed to 1.
- Similarly,  $\max(a, b)$  is an  $n$ -bit operation.

So, the fastest possible functions of two variables are  $\min$  and  $\max$ . Similarly fast is computing the minimum and maximum of several (more than two) real numbers. Therefore, we will choose these functions for our modeling-oriented computer.

Summarizing the above-given analysis, we can conclude that our computer will contain modules of two type:

- modules that compute functions of one variable;
- modules that compute  $\min$  and  $\max$  of two or several numbers.

**How to combine these modules?** We want to combine these modules in such a way that the resulting computations are as fast as possible. How can we estimate the computation time? This time is combined from the times of modules on different layers. In principle, different elementary modules take somewhat different time to perform: e.g., minimum is usually computed faster than, say, a square of a number. However, in most computers, all directly hardware supported operations usually take approximately the same time (the largest difference is in the order of 4). Therefore, to get an order-of-magnitude approximation of the computation time, we can assume that all elementary modules require the same computation time.

In this first, crude approximation, the total time required for an algorithm to be performed on a parallel machine is proportional to the number of sequential steps that it takes. We can describe this number of steps in clear geometric terms:

- at the beginning, the input numbers are processed by some processors; these processors form the *first layer* of computations;
- the results of this processing may then go into different processors, that form the *second layer*;
- the results of the second layer of processing go into the *third layer*,
- etc.

In these terms, the fewer layers the computer has, the faster it is:

- functions that can be computed by a single-layer computer require the smallest possible computation time, namely, the time  $\Delta t$  which is equal to the computation time of each processing module;
- functions which can be compare by 2-layer computers (with layers consisting of the above-described standard modules) require the computation time  $2 \cdot \Delta t$ ;
- functions which can be compare by 3-layer computers (with layers consisting of the above-described standard modules) require the computation time  $3 \cdot \Delta t$ , etc.

So, in the above-described first approximation, looking for the *fastest* model means looking for a combination of processors into the *smallest possible* number of layers.

**First result: in the first approximation, fuzzy rule based methodology is the fastest** [220]. It turns out that if we restrict ourselves to computers with one or two layers, then there exist functions that cannot be approximated. Actually, these non-approximable functions are not exotic at all: e.g. the function  $f(x_1, x_2) = x_1 + x_2$  on the domain  $[-1, 1]^2$  cannot be approximated by 2-layer computers even with accuracy  $\varepsilon = 0.4$ .

For *three* layers, we already get a universal approximation tool. The corresponding universal approximation result shows that for every real num-

bers  $T > 0$  and  $\varepsilon > 0$ , and for every continuous function  $f : [-T, T]^n \rightarrow R$ , there exists a function  $\tilde{f}$  that is  $\varepsilon$ -close to  $f$  on  $[-T, T]^n$  and that is computable on a 3-layer computer, or, more precisely, a function  $\tilde{f}$  of the type  $\max(A_1, \dots, A_k)$ , where  $A_j = \min(f_{j1}(x_1), \dots, f_{jn}(x_n))$ .

These functions can be explicitly described as the results of applying fuzzy rule based modeling methodology: Indeed, let us define

$$U = \max_{i,j,x_i \in [-T,T]} |f_{ji}(x_i)|,$$

and

$$\mu_{ji}(x_i) = \frac{f_{ji}(x_i) - (-U)}{U - (-U)}.$$

Let us now assume that the experts' rules base consists of exactly two rules:

- “if one of the conditions  $C_j$  is true, then  $u = U$ ”;
- “else,  $u = -U$ ”,

where each condition  $C_j$  means that the following  $n$  conditions are satisfied:

- $x_1$  satisfies the property  $C_{j1}$  (described by a membership function  $\mu_{j1}(x_1)$ );
- $x_2$  satisfies the property  $C_{j2}$  (described by a membership function  $\mu_{j2}(x_2)$ );
- ...
- $x_n$  satisfies the property  $C_{jn}$  (described by a membership function  $\mu_{jn}(x_n)$ ).

In logical terms, the condition  $C$  for  $u = U$  has the form

$$(C_{11} \& \dots \& C_{1n}) \vee \dots \vee (C_{k1} \& \dots \& C_{kn}).$$

If we use  $\&$  for  $\min$ , and  $\vee$  for  $\max$  (these are the simplest choices in intelligent modeling methodology), then the degree  $\mu_C$  with which we believe in a condition  $C = C_1 \vee \dots \vee C_k$  can be expressed as:

$$\mu_C = \max[\min(\mu_{11}(x_1), \dots, \mu_{1n}), \dots, \min(\mu_{k1}, \dots, \mu_{kn})].$$

Correspondingly, the truth value of a condition for  $u = -U$  is  $1 - \mu_C$ . According to fuzzy rule based modeling methodology, we must use a *defuzzification* to determine the actual output, which in this case leads to the choice of

$$u = \frac{U \cdot \mu_C + (-U) \cdot (1 - \mu_C)}{\mu_C + (1 - \mu_C)}.$$

Because of our choice of  $\mu_{ji}$ , one can easily see that this expression coincides exactly with the function  $\max(A_1, \dots, A_k)$ , where  $A_j = \min(f_{j1}(x_1), \dots, f_{jn}(x_n))$ . So, we get exactly the expressions that stem from the fuzzy rule based modeling methodology.

Let us summarize our logic:

- we are interested in the modeling methodologies that allow the fastest possible computations of the output  $u$  from the given inputs  $x_1, \dots, x_n$ ;
- the fastest computations are on a parallel computer with simple modules (which are explicitly described in the text);
- in the first approximation, each of the standard modules requires exactly the same computation time;
- in this approximation, the time to perform an algorithm is exactly proportional to the number of layers; so, minimizing computation time means exactly minimizing the total number of layers;
- it turns out that parallel computers with 1 or 2 layers are not enough to get the universal approximation property, and that 3 layers are already enough;
- we also show that functions  $u(x_1, \dots, x_n)$  computed by 3-layer computers are exactly the ones produced by the fuzzy rule based modeling methodology.

So (in the above approximation), *for modeling problems, the fastest possible universal computation scheme corresponds to using fuzzy rule based modeling methodology.*

**Second result: neural network modeling methodology is the fastest.** We have considered *digital* parallel computers. If we use *analog* processors instead, then min and max stop being the simplest functions. Instead, the sum is the simplest: if we just join the two wires together, then the resulting current is equal to the sum of the two input currents.

In this case, if we use a sum (and more general, linear combination) instead of min and max, 3-layer computers are also universal approximation tools; the corresponding computers correspond to *neural networks* (see, e.g., [219]).

### 7.3. CONCLUSIONS

The analysis given above leads us to the following qualitative conclusion:

- If our main objective is to design a model as fast as possible, then we should use fuzzy rule based modeling.
- If our main objective is *smoothness* of a model, then neural network modeling is preferable.
- If our main objective is *computational simplicity*, then:
  - for *software implementations*, fuzzy rule based modeling is better; and
  - for *hardware implementations*, neural network modeling is better.
- In control applications, if our main objective is *stability* of the resulting control, then fuzzy rule based modeling is preferable.

**Caution:** These conclusions are based on comparing the *potentials* of these two methods, without taking into consideration the imperfection of their implementations. Therefore, these conclusions should be taken as “fuzzy” *guidelines* rather than “crisp” preferences.

For example, the fact that fuzzy rule based modeling has a better potential in achieving stability than neural network modeling does not necessarily mean that the use of the actual fuzzy controller will always lead to more stability than the use of the actual neural network controller.

#### 7.4. COLLABORATE, NOT COMPETE

We have just seen that each of the two intelligent modeling methodologies (fuzzy rule based modeling and neural network modeling) has its own advantages. It is therefore desirable to *combine* these two methodologies into a single methodology that would use the advantages of both.

Depending on which of the methodologies we start with, we end up with two possible ways of combining these two methodologies:

- we can start with fuzzy rule based modeling methodology, and use a neural network to tune the rules and their parameters; such combined systems, called *neural fuzzy* models, have been described above;
- we can also start with the neural networks; then, the fuzzy inputs and fuzzy tuning rules will enhance the modeling capability of conventional neural networks.

These combined systems are a path to follow.

**Acknowledgments.** This paper was partly supported by the NSF grant EEC-9322370. We are thankful to numerous researchers, especially to Bernadette Bouchon-Meunier, Piero Bonissone, Jim Buckley, László Kóczy, Bart Kosko, Boris Kovalerchuk, Vera Kůrková, Thomas Runkler, David Sprecher, Li-Xin Wang, Ronald R. Yager, and Lotfi A. Zadeh, for valuable discussions and interesting preprints.

We are especially thankful to Didier Dubois and Henri Prade, the editors of the present Handbook, for their thorough editing and great help. Judging by the amount of thorough work they put into this chapter, we truly view them as our co-authors.

## References

### Fuzzy rule-based modeling methodology: an additional reference

- Larsen, P. M. (1980). Industrial applications of fuzzy logic control, *Internat. J. Man-Machine Stud.*, **12**, 3–10.

**Universal approximation results for the basic fuzzy rule based modeling methodology: basic results**

- Buckley, J. J. (1992). Universal Fuzzy Controllers, *Automatica*, **28**, 1245–1248.
- Buckley, J. J. (1993). Controllable processes and the fuzzy controller, *Fuzzy Sets and Systems*, **53**, 27–31.
- Buckley, J. J. (1993a). Sugeno Type Controllers are Universal Controllers, *Fuzzy Sets and Systems*, **53**, 299–303.
- Buckley, J. J. (1993d). Approximation paper: Part I, *Proceedings of the Third International Workshop on Neural Networks and Fuzzy Logic, Houston, TX, June 1–3, 1992*, NASA, January 1993, I (NASA Conference Publication No. 10111). 170–173.
- Buckley, J. J. (1993f). Applicability of the fuzzy controller, In: Wang, P. Z. and Loe, K. F. (eds.), *Advances in Fuzzy Systems: Application and Theory*, World Scientific, Singapore.
- Buckley, J. J. and Czogala, E. (1993e). Fuzzy models, fuzzy controllers, and neural nets, *Proc. Polish Academy of Sciences*.
- Buckley, J. J. and Hayashi, Y. (1993b). Fuzzy input-output controllers are universal approximators, *Fuzzy Sets and Systems*, **58**, 273–278.
- Buckley, J. J., Hayashi, Y., and Czogala, E. (1992a). On the equivalence of neural nets and fuzzy expert systems, *Proc. of Int. Joint Conf. on Neural Networks*, June 7–11, Baltimore, MD, **2**, 691–695.
- Buckley, J. J., Hayashi, Y., and Czogala, E. (1993c). On the equivalence of neural nets and fuzzy expert systems, *Fuzzy Sets and Systems*, **53**, 129–134.
- Cao, Z. (1990). Mathematical principle of fuzzy reasoning, *Proceedings NAFIPS'90*, Toronto, Canada, June 1990, 362–365.
- Cao, Z., Kandel, A., and Han, J. (1990a). Mechanism of fuzzy logic controller, *Proceedings of ISUMA'90*, Univ. of Maryland, December 1990, 603–607.
- Castro, J. L. (1995). Fuzzy logic controllers are universal approximators, *IEEE Trans. Syst., Man, Cybern.*, **25** (4), 629–635.
- Dubois, D., Grabisch, M., and Prade, H. (1992). Gradual rules and the approximation of functions, *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, July 17–22, 629–632.
- Hayashi, Y., Buckley, J. J., and Czogala, E. (1992). Fuzzy expert systems versus neural networks, *Proc. of Int. Joint Conf. on Neural Networks*, June 7–11, Baltimore, MD, **2**, 720–726.
- Hayashi, Y., Buckley, J. J., and Czogala, E. (1992a). Approximations between fuzzy expert systems and neural networks, *Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks*, July 17–22, Iizuka, Japan, 135–139.
- Jou, C.-C. (1992). On the mapping capabilities of fuzzy inference systems, *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, Maryland, June 7–11, **2**, 708–713.
- Kawamoto, S., Tada, K., Onoe, N., Ishigame, A., and Taniguchi, T. (1992). Construction of exact fuzzy system for nonlinear system and its stability analysis, *8th Fuzzy System Symposium*, Hiroshima, 517–520 (in Japanese).
- Kosko, B. (1991). *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*, Prentice Hall, 1991.
- Kosko, B. (1992). Fuzzy Systems as Universal Approximators, *IEEE Int. Conf. on Fuzzy Systems*, San Diego, CA, March 1992, 1143–1162.
- Kosko, B. (1992a). Fuzzy function approximation, *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, Maryland, June 7–11, **1**, 209–213.
- Kosko, B. (1994). Fuzzy Systems as Universal Approximators, *IEEE Trans. on Computers*, **43** (11), 1329–1333.
- Kosko, B. (1994a). Optimal fuzzy rules cover extrema, *Proceedings of the World Congress on Neural Networks WCNN'94*.

- Kosko, B. (1995). Optimal fuzzy rules cover extrema, *International Journal of Intelligent Systems*, **10** (2), 249–255.
- Kosko, B. (1996a). Additive fuzzy systems: from function approximation to learning, In: C. H. Chen (ed.), *Fuzzy Logic and Neural Network Handbook*, McGraw-Hill, N.Y., 9-1–9-22.
- Kosko, B. and Dickerson, J. A. (1995a). Function approximation with additive fuzzy systems, Chapter 12 in: Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y., 313–347.
- Nguyen, H. T. and Kreinovich, V. (1992a). *On approximation of controls by fuzzy systems*, Technical Report 92-93/302, LIFE Chair of Fuzzy Theory, Tokyo Institute of Technology.
- Nguyen, H. T. and Kreinovich, V. (1993). On Approximation of Controls by Fuzzy Systems, *Proceedings of Fifth IFSA Congress*, Seoul, Korea, **2**, 1414–1417.
- Wang, L.-X. (1992). Fuzzy Systems Are Universal Approximators, *Proceedings of Second IEEE International Conference on Fuzzy Systems*, San Diego, CA, March 1992, 1163–1170.
- Wang, L.-X. (1994). *Adaptive Fuzzy Systems and Control*, Prentice-Hall, Englewood-Cliffs, NJ.
- Wang, L.-X. and Mendel, J. M. (1991). *Generating fuzzy rules from numerical data with applications*, University of Southern California, Signal and Image Processing Institute, Technical Report USC-SIPI # 169.
- Wang, L.-X. and Mendel, J. M. (1992a). Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Transactions on Neural Networks*, **3**, 807–814.
- Wang, L.-X. and Mendel, J. M. (1992b). Generating fuzzy rules by learning from examples, *IEEE Transactions on Systems, Man, and Cybernetics*, **22**, 1414–1417.
- Yager, R. R. and Filev, D. P. (1994). *Essentials of fuzzy modeling and control*, J. Wiley & Sons.
- Ying, H. (1994). Sufficient conditions on general fuzzy systems as function approximators, *Automatica*, **30** (3), 521–525.
- Zeng, X.-J. and Singh, M. G. (1994). Approximation theory of fuzzy systems - SISO case, *IEEE Trans. on Fuzzy Systems*, **2** (2), 162–176.
- Zeng, X.-J. and Singh, M. G. (1995). Approximation theory of fuzzy systems - MIMO case, *IEEE Trans. on Fuzzy Systems*, **3**, 219–235.

#### **Takagi-Sugeno methodology, its generalizations, and the corresponding universal approximation results**

- Buckley, J. J. (1993a). Sugeno Type Controllers are Universal Controllers, *Fuzzy Sets and Systems*, **53**, 299–303.
- Buckley, J. J. (1993d). Approximation paper: Part I, *Proceedings of the Third International Workshop on Neural Networks and Fuzzy Logic, Houston, TX, June 1–3, 1992*, NASA, January 1993, **I** (NASA Conference Publication No. 10111). 170–173.
- Buckley, J. J. and Hayashi, Y. (1993b). Fuzzy input-output controllers are universal approximators, *Fuzzy Sets and Systems*, **58**, 273–278.
- Cao, S. G., Rees, N. W., and Feng, G. (1995). Fuzzy modelling and identification for a class of complex dynamic systems, *Proceedings of the Pacific-Asia Conference on Expert Systems*, Huanshan, China, 212–217.
- Cao, S. G., Rees, N. W., and Feng, G. (1996a). Analysis and design of uncertain fuzzy control systems. Part I: Fuzzy modelling and identification, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 640–646.
- Cao, S. G., Rees, N. W., and Feng, G. (1996b). Analysis and design of uncertain fuzzy control systems. Part II: Fuzzy controller design, *Proceedings of the Fifth IEEE In-*

- ternational Conference on Fuzzy Systems FUZZ-IEEE'96, New Orleans, September 8–11, **1**, 647–653.
- Chak, C. K., Feng, G., and Cao, S. G. (1996). Universal fuzzy controllers, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **3**, 2020–2025.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control, *IEEE Transactions on Systems, Man, and Cybernetics*, **15**, 116–132.

#### Universal approximation results for the ellipsoid approach

- Dickerson, J. A. and Kosko, B. (1993). Fuzzy function learning with covariant ellipsoids, *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, March 28–April 1, **3**, 1162–1167.
- Dickerson, J. A. and Kosko, B. (1993a). Fuzzy function approximation with supervised ellipsoidal learning, *Proceedings of the World Congress on Neural Networks*, Portland, Oregon, July 11–15, **2**, 9–17.
- Dickerson, J. A. and Kosko, B. (1996). Fuzzy function approximation with ellipsoidal rules, *IEEE Trans. Syst., Man, Cybern.*, **26** (4), 542–560.
- Kim, H. M. and Kosko, B. (1996). Fuzzy prediction and filtering in impulsive noise, *Fuzzy Sets and Systems*, **77**, 15–33.
- Kosko, B. (1994). Fuzzy Systems as Universal Approximators, *IEEE Trans. on Computers*, **43** (11), 1329–1333.
- Kosko, B. (1994a). Optimal fuzzy rules cover extrema, *Proceedings of the World Congress on Neural Networks WCNN'94*.
- Kosko, B. (1995). Optimal fuzzy rules cover extrema, *International Journal of Intelligent Systems*, **10** (2), 249–255.
- Kosko, B. (1996a). Additive fuzzy systems: from function approximation to learning. In: Chen, C. H. (ed.), *Fuzzy Logic and Neural Network Handbook*, McGraw-Hill, N.Y., 9-1–9-22.

#### Universal approximation results for the case when fuzzy if-then rules use a more complicated implication

- Dubois, D., Grabisch, M., and Prade, H. (1992). Gradual rules and the approximation of functions, *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, July 17–22, 629–632.
- Dubois, D., Grabisch, M., and Prade, H. (1993). Synthesis of real-valued mappings based on gradual rules and interpolative reasoning, *Proc. of the IJCAI'93 Fuzzy Logic in AI Workshop*, Chaméry, France, Aug. 28–Sep. 3, 29–40.
- Dubois, D., Prade, H., and Grabisch, M. (1995). Gradual rules and the approximation of control laws, In: Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y., 147–181.

#### Universal approximation results for the case when different grade of truth are assigned to different rules

- Dickerson, J. A. and Kosko, B. (1993). Fuzzy function learning with covariant ellipsoids, *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, March 28–April 1, **3**, 1162–1167.
- Dickerson, J. A. and Kosko, B. (1993a). Fuzzy function approximation with supervised ellipsoidal learning, *Proceedings of the World Congress on Neural Networks*, Portland, Oregon, July 11–15, **2**, 9–17.

- Dickerson, J. A. and Kosko, B. (1996). Fuzzy function approximation with ellipsoidal rules, *IEEE Trans. Syst., Man, Cybern.*, **26** (4), 542–560.
- Kosko, B. (1992). Fuzzy Systems as Universal Approximators, *IEEE Int. Conf. on Fuzzy Systems*, San Diego, CA, March 1992, 1143–1162.
- Kosko, B. (1996a). Additive fuzzy systems: from function approximation to learning, In: Chen, C. H. (ed.), *Fuzzy Logic and Neural Network Handbook*, McGraw-Hill, N.Y., 9-1–9-22.
- Kosko, B. and Dickerson, J. A. (1995a). Function approximation with additive fuzzy systems, Chapter 12 in: Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y., 313–347.

#### Universal approximation results for hierarchical systems

- Buckley, J. J. (1993). Controllable processes and the fuzzy controller, *Fuzzy Sets and Systems*, **53**, 27–31.
- Buckley, J. J., and Hayashi, Y. (1993b). Fuzzy input-output controllers are universal approximators, *Fuzzy Sets and Systems*, **58**, 273–278.
- Buckley, J. J., Hayashi, Y., and Czogala, E. (1992a). On the equivalence of neural nets and fuzzy expert systems, *Proc. of Int. Joint Conf. on Neural Networks*, June 7–11, Baltimore, MD, **2**, 691–695.
- Buckley, J. J., Hayashi, Y., and Czogala, E. (1993c). On the equivalence of neural nets and fuzzy expert systems, *Fuzzy Sets and Systems*, **53**, 129–134.
- Hayashi, Y. and Buckley, J. J. (1994). Approximations between fuzzy expert systems and neural networks, *International Journal of Approximate Reasoning*, **10** (1), 63–73.
- Hayashi, Y., Buckley, J. J., and Czogala, E. (1992). Fuzzy expert systems versus neural networks, *Proc. of Int. Joint Conf. on Neural Networks*, June 7–11, Baltimore, MD, **2**, 720–726.
- Hayashi, Y., Buckley, J. J., and Czogala, E. (1992a). Approximations between fuzzy expert systems and neural networks, *Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks*, July 17–22, Iizuka, Japan, 135–139.
- Jang, J.-S. R. and Sun, C.-T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems, *IEEE Transactions on Neural Networks*, **4**, 156–159.
- Kosko, B. (1995b). Combining fuzzy systems, *Proceedings of the IEEE International Conference on Fuzzy Systems FUZZ-IEEE/IFES'95*, March 1995, **4**, 1855–1863.

#### Universal approximation results for simplified versions of fuzzy rule based modeling methodology

- Buckley, J. J. (1993a). Sugeno Type Controllers are Universal Controllers, *Fuzzy Sets and Systems*, **53**, 299–303.
- Buckley, J. J. (1993d). Approximation paper: Part I, *Proceedings of the Third International Workshop on Neural Networks and Fuzzy Logic*, Houston, TX, June 1–3, 1992, NASA, January 1993, **I** (NASA Conference Publication No. 10111), 170–173.
- El Hajjaji, A. and Rachid, A. (1994). Explicit formulas for fuzzy controller, *Fuzzy Sets and Systems*, **62**, 135–141.
- Kóczy, L. T. and Tikk, D. (1996). Approximation in rule bases, *IPMU'96: Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, July 1–5, 1996*, 489–494.
- Su, C.-Y. and Stepanenko, Y. (1994). Adaptive control of a class of nonlinear systems with fuzzy logic, *Proceedings of the Third IEEE International Conference on Fuzzy Systems FUZZ-IEEE'94*, Orlando, FL, June 26–29, **2**, 779–785.
- Su, C.-Y. and Stepanenko, Y. (1994a). Adaptive control of a class of nonlinear systems with fuzzy logic, *IEEE Transactions on Fuzzy Systems*, **2** (4), 285–294.

- Tan, S., and Vandewalle, J. (1996). Defuzzification, structure transparency, and fuzzy system learning, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 470–474.
- Zeng, X.-J. and Singh, M. G. (1996). Approximation accuracy analysis of fuzzy systems as function approximators, *IEEE Trans. on Fuzzy Systems*, **4**, 44–63.

**Universal approximation results for distributed systems (and for universal controllers)**

- Cao, S. G., Rees, N. W., and Feng, G. (1996). Fuzzy control of nonlinear discrete-time systems, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 265–271.
- Kreinovich, V., Nguyen, H. T., and Sirisaengtaksin, O. (1994a). On approximations of controls in distributed systems by fuzzy controllers, *Proceedings of the 5th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'94*, Paris, July 4–8, **1**, 79–83.
- Kreinovich, V., Nguyen, H. T., and Sirisaengtaksin, O. (1995a). On approximation of controls in distributed systems by fuzzy controllers, In: Bouchon-Meunier, B., Yager, R. R., and Zadeh, L. A. (eds.), *Fuzzy Logic and Soft Computing*, World Scientific, 137–145.
- Nguyen, H. T., Kreinovich, V., and Sirisaengtaksin, O. (1996). Fuzzy control as a universal control tool, *Fuzzy Sets and Systems*, **80** (1), 71–86.

**Universal approximation results for discrete systems: Application to expert system design**

- Garey, M. and Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco.
- Kreinovich, V. (1987). Semantics of S. Yu. Maslov's iterative method, *Problems of Cybernetics*, Moscow, 1987, **131**, 30–62 (in Russian); English translation in: Kreinovich, V. and Mints, G. (eds.), *Problems of reducing the exhaustive search*, American Mathematical Society, Providence, RI, 1996.
- Kreinovich, V. (1996). S. Maslov's Iterative Method: 15 Years Later (Freedom of Choice, Neural Networks, Numerical Optimization, Uncertainty Reasoning, and Chemical Computing). a chapter in Kreinovich, V. and Mints, G. (eds.), *Problems of reducing the exhaustive search*, American Mathematical Society, Providence, RI.
- Kreinovich, V. and Fuentes, L. O. (1991). Simulation of chemical kinetics - a promising approach to inference engines, in: Liebowitz, J. (ed.), *Proceedings of the World Congress on Expert Systems, Orlando, Florida, 1991*, Pergamon Press, N.Y., **3**, 1510–1517.
- Kreinovich, V. and Mints, G. (eds.) (1996a). *Problems of reducing the exhaustive search*, American Mathematical Society (AMS Translations — Series 2), **178**, Providence, RI.
- Maslov, S. Yu. (1981). Iterative methods in intractable problems as a model of intuitive methods, *Abstracts of the 9th All-Union Symposium on Cybernetics*, 52–56 (in Russian).
- Maslov, S. Yu. (1983). Asymmetry of cognitive mechanisms and its implications, *Semiotika i Informatika*, **20**, 3–31 (in Russian).
- Maslov, S. Yu. (1987). *Theory of Deductive Systems and its Applications*, MIT Press, Cambridge, MA.
- Sirisaengtaksin, O., Fuentes, L. O., and Kreinovich, V. (1995). Non-traditional neural networks that solve one more intractable problem: propositional satisfiability, *Proceedings of the First International Conference on Neural, Parallel, and Scientific Computations*, Atlanta, GA, May 28–31, 1995, **1**, 427–430.

**Universal approximation results that describe the possibility of  
approximating a smooth system by a smooth one, and a simple system  
by a simple one**

- Fantuzzi, C. and Rovatti, R. (1996). On the approximation capabilities of the homogeneous Takagi-Sugeno model, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **2**, 1067–1072.
- Galichet, S. and Foulloy, L. (1993). Fuzzy equivalence of classical controllers, *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies*, Aachen, Germany, **3**, 1567–1573.
- Gordon, W. J. and Riesenfeld, R. F. (1974). B-spline curves and surfaces, In: Barnhill, R. E. and Riesenfeld, R. F. (eds.), *Computer Aided Geometric Design*, Academic Press, N.Y.
- Kovalerchuk, B. (1996). Second interpolation in fuzzy control, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 150–155.
- Kovalerchuk, B. (1996a). *Fuzzy control with the second interpolation*, Working paper, Louisiana State University, Baton Rouge, LA, 1996.
- Kovalerchuk, B. and Yusupov, H. (1993). Fuzzy control as interpolation, *Proceedings of the 5th IFSA Congress*, Seoul, Korea, July 4–9, 1151–1154.
- Kovalerchuk, B., Yusupov, H., and Kovalerchuk, N. (1994). Comparison of interpolations in fuzzy control, *Proceedings of the 2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control*, Lund, Sweden, 76–81.
- Kreinovich, V., Quintana, C., Lea, R., Fuentes, O., Lokshin, A., Kumar, S., Boricheva, I., and Reznik, L. (1992). What non-linearity to choose? Mathematical foundations of fuzzy control, *Proceedings of the 1992 International Conference on Fuzzy Systems and Intelligent Control*, Louisville, KY, 349–412.
- Meyer-Gramann, K. D. (1993). Easy implementation of fuzzy controller with a smooth control surface, *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies*, Aachen, Germany, **1**, 117–123.
- Mohler, R. R. (1991). *Nonlinear systems. 1. Dynamics and control*, Prentice Hall, Englewood Cliff, NJ.
- Raymond, C., Boverie, S., and Le, J. M. (1993). Practical realization of fuzzy controllers; comparison with conventional methods, *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies*, Aachen, Germany, **1**, 149–155.
- Rovatti, R. (1996). Takagi-Sugeno models as approximators in Sobolev norms: the SISO case, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **2**, 1060–1066.
- Runkler, T. A. and Glesner, M. (1993). A new approach to fuzzy logic controller realizations using B-splines, *Proceedings of the International Workshop on Current Issues in Fuzzy technology*, Trento, Italy.
- Runkler, T. A. and Glesner, M. (1993). Approximative Synthese von Fuzzy-Controllern in Fuzzy Logic, In: Reusch, B. (ed.), *Fuzzy Logic — Theorie und Praxis*, Springer-Verlag, Berlin, 22–31.
- Tilli, T. A. (1993). Practical tools for simulation and optimization fuzzy systems with various operators and defuzzification methods, *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies*, Aachen, Germany, **1**, 256–262.
- Zhang, J. and Knoll, A. (1996). Constructing fuzzy controllers with B-spline models, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 416–421.
- Zhang, J., Raczkowsky, J., and Herp, A. (1994). Emulation of spline curves and its application in robot motion control, *Proceedings of the IEEE International Conference on Fuzzy Systems FUZZ-IEEE'94*, Orlando, FL, 831–836.
- Zhang, J., Wille, F., and Knoll, A. (1996a). Modular design of fuzzy controller integrating deliberate and reactive strategies, *Proceedings of the IEEE International Conference*

on Robotics and Automation.

**Universal approximation results that describe the possibility of approximating, for each controlled system, a stabilizing control by stabilizing controls**

- Buckley, J. J. (1992). Universal Fuzzy Controllers, *Automatica*, **28**, 1245–1248.
- Buckley, J. J. (1993). Controllable processes and the fuzzy controller, *Fuzzy Sets and Systems*, **53**, 27–31.
- Buckley, J. J. (1993f). Applicability of the fuzzy controller, In: Wang, P.-Z. and Loe, K. F. (eds.), *Advances in Fuzzy Systems: Application and Theory*, World Scientific, Singapore.
- Buckley, J. J. (1995). System stability and the fuzzy controller, In: Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley and Sons, N.Y., 51–63.
- Cao, S. G., Rees, N. W., and Feng, G. (1996). Fuzzy control of nonlinear discrete-time systems, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 265–271.
- Chak, C. K., Feng, G., and Cao, S. G. (1996). Universal fuzzy controllers, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **3**, 2020–2025.
- Jagannathan, S. (1996). Adaptive discrete-time fuzzy logic control of feedback linearizable nonlinear systems, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **2**, 1273–1278.
- Lei, S. and Langari, R. (1996). An approach to synthesis and approximation of stable fuzzy logic controllers, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **2**, 1446–1452.
- Wang, L.-X. (1993). Stable adaptive fuzzy control of nonlinear systems, *IEEE Trans. Fuzzy Syst.*, **1**, 146–155.

**Universal approximation results for the case when we have both fuzzy if-then rules describing a model and fuzzy if-then rules describing control**

- Abello, J., Kreinovich, V., Nguyen, H. T., Sudarsky, S., and Yen, J. (1994). Computing an appropriate control strategy based only on a given plant's rule-based model is NP-hard, In: Hall, L., Ying, H., Langari, R., and Yen, J. (eds.), *NAFIPS/IFIS/NASA '94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, 331–332.
- Bouchon-Meunier, B., Kreinovich, V., Lokshin, A., and Nguyen, H. T. (1996). On the formulation of optimization under elastic constraints (with control in mind). *Fuzzy Sets and Systems*, **81** (1), 5–29.
- Chen, G., Pham, T. T., and Weiss, J. J. (1995). Fuzzy modeling of control systems, *IEEE Transactions on Aerospace and Electronic Systems*, **31** (1), 414–429.
- Nguyen, H. T., Kreinovich, V., and Sirisaengtaksin, O. (1996). Fuzzy control as a universal control tool, *Fuzzy Sets and Systems*, **80** (1), 71–86.
- Pham, T. T., Weiss, J. J., and Chen, G. R. (1992). Optimal fuzzy logic control for docking a boat, *Proc. of the Second International Workshop on Industrial Fuzzy Control and Intelligent Systems, Dec. 2–4, 1992*, College Station, TX, 66–73.

**Universal approximation results for the case when expert rules use unusual logical connectives**

- Kreinovich, V., Nguyen, H. T., and Walker, E. A. (1996b). Maximum entropy (MaxEnt) method in expert systems and intelligent control: new possibilities and limitations, In: Hanson, K. M. and Silver, R. N. (eds.), *Maximum Entropy and Bayesian Methods*, Kluwer Academic Publishers, Dordrecht, 93–100.
- Nguyen, H. T., and Kreinovich, V. (1997). Kolmogorov's Theorem and its impact on soft computing, In: Yager, R. R. and Kacprzyk, J. *The Ordered Weighted Averaging Operators: Theory, Methodology, and Applications*, Kluwer, Norwell, MA, 3–17.
- Nguyen, H. T., Kreinovich, V., and Sprecher, D. (1996a). Normal forms for fuzzy logic – an application of Kolmogorov's theorem *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, 4 (4), 331–349.
- Türkşen, I. B. and Kreinovich, V. (1996). Fuzzy implication revisited: a new type of fuzzy implication explains Yager's implication operation, In: Dimitrov, V. and Dimitrov, J. (eds.), *Fuzzy Logic and the Management of Complexity (Proceedings of the 1996 International Discourse)*, UTS Publ., Sydney, Australia, 3, 292–295.

### Universal approximation results for neural networks

- Blum, E. K. and Li, L. K. (1991). Approximation Theory and feedforward networks, *Neural Networks*, 4, 511–515.
- Cotter, N. E. (1990). The Stone-Weierstrass Theorem and Its Applications to Neural Networks, *IEEE Trans. on Neural Networks*, 1 (4), 290–295.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function, *Math. of Control, Signals, and Systems*, 2, 303–314, 1989.
- Funahashi, K. (1989). On the Approximate Realization of Continuous Mappings by Neural Networks, *Neural Networks*, 2, 183–192.
- Funahashi, K. and Nakamura, Y. (1993). Approximation of Dynamical Systems by continuous time recurrent neural networks, *Neural Networks*, 6, 801–806.
- Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, New York, NY.
- Hecht-Nielsen, R. (1987). Kolmogorov's Mapping Neural Network Existence Theorem, *Proceedings of First IEEE International Conference on Neural Networks*, 11–14, San Diego, CA.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer Feedforward Neural Networks Are Universal Approximators, *Neural Networks*, 2, 359–366.
- Hornik, K. (1991). Approximation Capabilities of Multilayer Feedforward Neural Networks, *Neural Networks*, 4, 251–257.
- Ito, Y. (1991). Approximation of functions on a compact set by finite sums of a sigmoid function without scaling, *Neural Networks*, 4, 817–826.
- Ito, Y. (1992). Approximation of continuous functions on  $R^d$  by linear combinations of shifted rotations of a sigmoid function with and without scaling, *Neural Networks*, 5, 105–115.
- Kosko, B. (1992b). *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ.
- Kreinovich, V. (1991b). Arbitrary Nonlinearity is Sufficient to Represent All Functions by Neural Networks: A Theorem, *Neural Networks*, 4, 381–383.
- Kreinovich, V. and Quintana, C. (1991a). Neural networks: what non-linearity to choose?, *Proceedings of the 4th University of New Brunswick Artificial Intelligence Workshop*, Fredericton, N.B., Canada, 627–637.
- Kreinovich, V. and Sirisaengtaksin, O. (1993). 3-layer neural networks are universal approximators for functionals and for control strategies, *Neural, Parallel, and Scientific Computations*, 1, 325–346.
- Kreinovich, V., Sirisaengtaksin, O., and Nguyen, N. T. (1995). Sigmoid neurons are the safest against additive errors, *Proceedings of the First International Conference on Neural, Parallel, and Scientific Computations*, Atlanta, GA, May 28–31, 1, 419–423.

- Kůrková, V. (1991). Kolmogorov's Theorem is Relevant, *Neural Computation*, **3**, 617–622.
- Kůrková, V. (1992). Kolmogorov's Theorem and Multilayer Neural Networks, *Neural Networks*, **5**, 501–506.
- Kůrková, V. (1992a). Universal approximation using feedforward neural networks with Gaussian bar units, *Proceedings of ECAI'92, European Conference on Artificial Intelligence*, Wiley, Chichester, 193–197.
- Leshno, M., Lin, V. Ya., Pinkus, A., and Schocken, S. (1993). Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function, *Neural Networks*, **6**, 861–867.
- Nakamura, M., Mines, R., and Kreinovich, V. (1993). Guaranteed intervals for Kolmogorov's theorem (and their possible relation to neural networks). *Interval Computations*, No. 3, 183–199.
- Park, J. and Sandberg, J. W. (1991). Universal approximation using radial basis function networks, *Neural Computation*, **3**, 246–257.
- Sirisaengtaksin, O. and Kreinovich, V. (1993). Neural networks that are not sensitive to the imprecision of hardware neurons, *Interval Computations*, No. 4, p. 100–113.
- Wray, J. and Green, G. G. R. (1995). Neural networks, approximation theory, and finite precision computation, *Neural Networks*, **8** (1), 31–37.

#### Universal approximation results for fuzzy neural networks

- Buckley, J. J. and Hayashi, Y. (1994). Can fuzzy neural networks approximate continuous fuzzy functions?, *Fuzzy Sets and Systems*, **61**, 43–52.
- Buckley, J. J. and Hayashi, Y. (1994a). Hybrid fuzzy neural networks are universal approximators, *Proceedings of the Third IEEE International Conference on Fuzzy Systems FUZZ-IEEE'94*, Orlando, FL, June 26–29, **1**, 238–243.
- Feuring, T. and Lippe, W.-M. (1995). Fuzzy neural networks are universal approximators, *Proceedings of the IFSA World Congress*, San Paulo, Brazil, 659–662.
- Hayashi, Y. and Buckley, J. J. (1994). Approximations between fuzzy expert systems and neural networks, *International Journal of Approximate Reasoning*, **10** (1), 63–73.

#### Universal approximation results for systems with fuzzy inputs and interval inputs

- Buckley, J. J. (1991). Fuzzy I/O controller, *Fuzzy Sets and Systems*, **43**, 127–137.
- Buckley, J. J. and Hayashi, Y. (1993b). Fuzzy input-output controllers are universal approximators, *Fuzzy Sets and Systems*, **58**, 273–278.
- Kreinovich, V. and Nguyen, H. T. (1994b). Applications of fuzzy intervals: a skeletal outline, In: Hall, L., Ying, H., Langari, R., and Yen, J. (eds.), *NAFIPS/IFIS/NASA '94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, 461–463.
- Kreinovich, V. and Nguyen, H. T. (1995b). On Hilbert's Thirteenth Problem for Soft Computing, *Proceedings of the Joint 4th IEEE Conference on Fuzzy Systems and 2nd IFES*, Yokohama, Japan, March 20–24, 1995, **IV**, 2089–2094.
- Lea, R., Kreinovich, V., and Trejo, R. (1996). Optimal interval enclosures for fractionally-linear functions, and their application to intelligent control, *Reliable Computing*, **2** (3), 265–286.
- Mouzouris, G. C. and Mendel, J. M. (1994). Non-Singleton Fuzzy Logic Systems, *Proceedings of Third IEEE International Conference on Fuzzy Systems*, June 1994, **1**, 456–461.

- Mouzouris, G. C. and Mendel, J. M. (1994a). *Non-Singleton Fuzzy Logic Systems: Theory and Application*, University of Southern California, Signal and Image Processing Institute, Report No. 262.
- Mouzouris, G. C. and Mendel, J. M. (1995). Nonlinear Time Series Analysis with Non-Singleton Fuzzy Logic Systems, *Proceedings IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, New York, NY, April 1995, 47–56.
- Mouzouris, G. C. and Mendel, J. M. (1996). Nonlinear predictive modeling using dynamic non-singleton fuzzy logic systems, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **2**, 1217–1223.
- Mouzouris, G. C. and Mendel, J. M. (1996a). Designing fuzzy logic systems for uncertain environments using a singular-value-QR decomposition method, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **1**, 295–301.
- Mouzouris, G. C. and Mendel, J. M. (1997). Non-Singleton Fuzzy Logic Systems: Theory and Application, *IEEE Transactions on Fuzzy Systems*, **5** (1), 56–71.
- Nakamura, M. (1994). What fuzzy interval operations should be hardware supported?, In: Hall, L., Ying, H., Langari, R., and Yen, J. (eds.), *NAFIPS/IFIS/NASA'94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, 393–397.
- Nguyen, H. T. and Kreinovich, V. (1995). Towards theoretical foundations of soft computing applications (1995). *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, **3** (3), 341–373.
- Nguyen, H. T., Kreinovich, V., Nesterov, V., and Nakamura, M. (1997a). On hardware support for interval computations and for soft computing: a theorem, *IEEE Transactions on Fuzzy Systems*, **5** (1), 108–127.
- Wu, K. C. (1994). A robot must be better than a human driver: an application of fuzzy intervals. In: Hall, L., Ying, H., Langari, R., and Yen, J. (eds.), *NAFIPS/IFIS/NASA'94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, 171–174.
- Wu, K. C. (1996). Fuzzy interval control of mobile robots, *Computers and Electrical Engineering*, **22** (3), 211–229.

#### Approximation with fewer rules

- Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Transactions on Information Theory*, **39**, 930–945.
- Bauer, P., Klement, E. P., Moser, B., and Leikermoser, A. (1995). Modeling of fuzzy functions by fuzzy controllers, In: Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y.
- DeVore, R., Howard, R., and Micchelli, C. A. (1989). Optimal nonlinear approximation, *Manuscripta Mathematica*, **63**, 469–478.
- Dickerson, J. A., and Kosko, B. (1996). Fuzzy function approximation with ellipsoidal rules, *IEEE Trans. Syst., Man, Cybern.*, **26** (4), 542–560.
- Foo, N. (1994). New perspectives and old problems ..., *Proceedings of the Soft Computing Symposium*, University of New South Wales.
- Jones, L. K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training, *The Annals of Statistics*, **20**, 601–613.

- Kóczy, L. T. and Tikk, D. (1996). Approximation in rule bases, *IPMU'96: Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, July 1-5, 1996*, 489-494.
- Kosko, B. (1994). Fuzzy Systems as Universal Approximators, *IEEE Trans. on Computers*, **43** (11), 1329-1333.
- Kosko, B. (1994a). Optimal fuzzy rules cover extrema, *Proceedings of the World Congress on Neural Networks WCNN'94*.
- Kosko, B. (1995). Optimal fuzzy rules cover extrema, *International Journal of Intelligent Systems*, **10** (2), 249-255.
- Kosko, B. (1996). Asymptotic stability of feedback additive fuzzy systems, *Proceedings of the World Congress on Neural Networks WCNN'96*, September 1996.
- Kosko, B. (1996a). Additive fuzzy systems: from function approximation to learning, In: Chen, C. H. (ed.), *Fuzzy Logic and Neural Network Handbook*, McGraw-Hill, N.Y., 9-1-9-22.
- Kosko, B. and Dickerson, J. A. (1995a). Function approximation with additive fuzzy systems, Chapter 12 in: Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y., 313-347.
- Kreinovich, V., Sirisaengtaksin, O., and Cabrera, S. (1994c). Wavelet neural networks are optimal approximators for functions of one variable, *Proceedings of the IEEE International Conference on Neural Networks*, Orlando, FL, July 1994, **1**, 299-303.
- Kürková, V., Kainen, P. C., and Kreinovich, V. (1995). Dimension-independent rates of approximation by neural networks and variation with respect to half-spaces, *Proceedings of World Congress on Neural Networks, WCNN'95, Washington, DC, July 1995*, INNS Press, NJ, **1**, 54-57.
- Kürková, V., Kainen, P. C., and Kreinovich, V. (1997). Estimates of the Number of Hidden Units and Variation with Respect to Half-spaces, *Neural Networks*, **10** (6), 1061-1068.
- Lee, J. and Chae, S. (1993). Completeness of fuzzy controller carrying a mapping  $f : R \rightarrow R$ , *Proceedings of the IEEE International Conference on Fuzzy Systems FUZZ-IEEE'93*, **1**, 231-235.
- Lee, J. and Chae, S. (1993a). Analysis of function duplicating capabilities of fuzzy controllers, *Fuzzy Sets and Systems*, **56**, 127-143.
- Lorentz, G. G. (1966). *Approximation of Functions*, Holt, Reinhart and Winston, New York, 1966.
- Watkins, F. A. (1994). *Fuzzy Engineering*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of California at Irvine.
- Watkins, F. A. (1995). The representation problem for additive fuzzy systems, *Proceedings of the IEEE International Conference on Fuzzy Systems FUZZ-IEEE/IFES'95*, March 1995, **1**, 117-122.
- Ying, H. (1994). Sufficient conditions on general fuzzy systems as function approximators, *Automatica*, **30** (3), 521-525.
- Zeng, X.-J. and Singh, M. G. (1994). Approximation theory of fuzzy systems - SISO case, *IEEE Trans. on Fuzzy Systems*, **2** (2), 162-176.
- Zeng, X. J. and Singh, M. G. (1995a). Approximation accuracy analysis of fuzzy systems with the center-average defuzzifier, *Proceedings Fourth IEEE Int. Conf. on Fuzzy Systems*, 109-116, Yokohama, March 1995.
- Zeng, X.-J. and Singh, M. G. (1996). Approximation accuracy analysis of fuzzy systems as function approximators, *IEEE Trans. on Fuzzy Systems*, **4**, 44-63.

#### How to choose the best variant of fuzzy ruled based modeling methodology

- Bouchon-Meunier, B., Kreinovich, V., Lokshin, A., and Nguyen, H. T. (1996). On the formulation of optimization under elastic constraints (with control in mind). *Fuzzy Sets and Systems*, **81** (1), 5-29.

- Kosheleva, O. (1996). Why Sinc? Signal Processing Helps Fuzzy Control, *SC-COSMIC, South Central Computational Sciences in Minority Institutions Consortium, Second Student Conference in Computational Sciences*, October 25–27, El Paso, TX, Abstracts, 19–21.
- Kreinovich, V. (1997). Random sets unify, explain, and aid known uncertainty methods in expert systems, in Goutsias, J., Mahler, R. P. S., and Nguyen, H. T. (eds.), *Random Sets: Theory and Applications*, Springer-Verlag, N.Y., 321–345.
- Kreinovich, V. (1996c). Maximum entropy and interval computations, *Reliable Computing*, **2** (1), 63–79.
- Kreinovich, V., Nguyen, H. T., and Walker, E. A. (1996b). Maximum entropy (MaxEnt) method in expert systems and intelligent control: new possibilities and limitations, In: Hanson K. M. and Silver, R. N. (eds.), *Maximum Entropy and Bayesian Methods*, Kluwer Academic Publishers, Dordrecht, 93–100.
- Kreinovich, V., Quintana, C., and Lea, R. (1991c). What procedure to choose while designing a fuzzy control? Towards mathematical foundations of fuzzy control, *Working Notes of the 1st International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, TX, 123–130.
- Kreinovich, V., Quintana, C., Lea, R., Fuentes, O., Lokshin, A., Kumar, S., Boricheva, I., and Reznik, L. (1992). What non-linearity to choose? Mathematical foundations of fuzzy control, *Proceedings of the 1992 International Conference on Fuzzy Systems and Intelligent Control*, Louisville, KY, 349–412.
- Kreinovich, V. and Tolbert, D. (1994d). Minimizing computational complexity as a criterion for choosing fuzzy rules and neural activation functions in intelligent control. In: Jamshidi, M., Nguyen, C., Lumia, R., and Yuh, J. (eds.), *Intelligent Automation and Soft Computing. Trends in Research, Development, and Applications. Proceedings of the First World Automation Congress (WAC'94). August 14–17, 1994, Maui, Hawaii*, TSI Press, Albuquerque, NM, **1**, 545–550.
- Mitaim, S. and Kosko, B. (1996). What is the best shape for a fuzzy set in function approximation?, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, New Orleans, September 8–11, **2**, 1237–1243.
- Mitaim, S. and Kosko, B. (1996a). Fuzzy function approximation and the shape of fuzzy sets, *Proceedings of the World Congress on Neural Networks WCNN'96*, September 1996.
- Nguyen, H. T. and Kreinovich, V. (1995). Towards theoretical foundations of soft computing applications (1995). *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, **3** (3), 341–373.
- Nguyen, H. T., Kreinovich, V., Lea, B., and Tolbert, D. (1992). How to control if even experts are not sure: robust fuzzy control, *Proceedings of the Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, December 2–4, 153–162.
- Nguyen, H. T., Kreinovich, V., Lea, B., and Tolbert, D. (1995a). Interpolation that leads to the narrowest intervals, and its application to expert systems and intelligent control, *Reliable Computing*, 1995, **3** (1), 299–316.
- Nguyen, H. T., Kreinovich, V., and Tolbert, D. (1993a). On robustness of fuzzy logics. *Proceedings of IEEE-FUZZ International Conference*, San Francisco, CA, March 1993, **1**, 543–547.
- Nguyen, H. T., Kreinovich, V., and Tolbert, D. (1994). A measure of average sensitivity for fuzzy logics, *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, **2** (4), 361–375.
- Nguyen, H. T. and Walker, E. A. (1997b). *A first course in fuzzy logic*, CRC Press, Boca Raton, FL.
- Ramer, A. and Kreinovich, V. (1992). Maximum entropy approach to fuzzy control, *Proceedings of the Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, December 2–4, 113–117.
- Ramer, A. and Kreinovich, V. (1994). Information complexity and fuzzy control, Chapter

- 4 in: Kandel, A. and Langholtz, G. (eds.), *Fuzzy Control Systems*, CRC Press, Boca Raton, FL, 75–97.
- Ramer, A. and Kreinovich, V. (1994a). Maximum entropy approach to fuzzy control, *Information Sciences*, **81** (3–4), 235–260.
- Smith, M. H. and Kreinovich, V. (1995). Optimal strategy of switching reasoning methods in fuzzy control, Chapter 6 in Nguyen, H. T., Sugeno, M., Tong, R., and Yager, R. (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y., 117–146.
- Zeng, X.-J. and Singh, M. G. (1996). Approximation accuracy analysis of fuzzy systems as function approximators, *IEEE Trans. on Fuzzy Systems*, **4**, 44–63.

**Comparison between fuzzy rules based modeling methodology and  
neural network modeling methodology**

- Kreinovich, V. and Bernat, A. (1994). Parallel algorithms for interval computations: an introduction, *Interval Computations*, No. 3, 6–62.
- Lea, R. N. and Kreinovich, V. (1995). Intelligent Control Makes Sense Even Without Expert Knowledge: an Explanation, *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), 140–145.