Fuzzy Logic and Neuro-fuzzy Systems: A Systematic Introduction

Yue Wu *wuyue@enjoyor.cc*
Enjoyor Inc
Hangzhou, 310030, China

Biaobiao Zhang *zhangbb@enjoyor.net*
Enjoyor Inc
Hangzhou, 310030, China

Jiabin Lu *lujiabin@gdut.edu.cn*
Faculty of Electromechanical Engineering
Guangdong University of Technology
Guangzhou, 510006, China

K. -L. Du *kldu@ece.concordia.ca*
Department of Electrical and Computer Engineering
Concordia University
Montreal, H3G 1M8, Canada
and
Enjoyor Inc
Hangzhou, 310030, China

Abstract

Fuzzy logic is a rigorous mathematical field, and it provides an effective vehicle for modeling the uncertainty in human reasoning. In fuzzy logic, the knowledge of experts is modeled by linguistic rules represented in the form of IF-THEN logic. Like neural network models such as the multilayer perceptron (MLP) and the radial basis function network (RBFN), some fuzzy inference systems (FISs) have the capability of universal approximation. Fuzzy logic can be used in most areas where neural networks are applicable. In this paper, we first give an introduction to fuzzy sets and logic. We then make a comparison between FISs and some neural network models. Rule extraction from trained neural networks or numerical data is then described. We finally introduce the synergy of neural and fuzzy systems, and describe some neuro-fuzzy models as well. Some circuits implementations of neuro-fuzzy systems are also introduced. Examples are given to illustrate the concepts of neuro-fuzzy systems.

Keywords: Fuzzy Set, Fuzzy Logic, Fuzzy Inference System, Neuro-fuzzy System, Neural Network, Mamdani Model, Takagi-Sugeno-Kang Model.

1. INTRODUCTION

Fuzzy set, a concept first proposed by Zadeh [123], is a method for modeling the uncertainty in human reasoning. Fuzzy logic is suitable for the representation of vague data and concepts on an intuitive basis, such as human linguistic description, e.g. the expressions *approximately*, *large*, *young*. The conventional set, also called the crisp set, can be treated as a special form of fuzzy set. Unlike the binary logic, fuzzy logic uses the notion of membership. A fuzzy set is uniquely determined by its membership function (MF), and it is also associated with a linguistically meaningful term.

Fuzzy logic provides a systematic tool to incorporate human experience. It is based on three core concepts, namely, fuzzy sets, linguistic variables, and possibility distributions. Fuzzy set is used to characterize linguistic variables whose values can be described qualitatively using a linguistic expression and quantitatively using an MF [124]. Linguistic expressions are useful for communicating concepts and knowledge with human beings, whereas MFs are useful for processing numeric input data. When a fuzzy set is assigned to a linguistic variable, it imposes an elastic constraint, called a possibility distribution, on the possible values of the variable.

Fuzzy logic is a rigorous mathematical discipline. Fuzzy reasoning is a straightforward formalism for encoding human knowledge or common sense in a numerical framework, and FISs can approximate arbitrarily well any continuous function on a compact domain [55], [113]. FISs and feedforward neural networks (FNNs) can approximate each other to any degree of accuracy [13]. Fuzzy logic first found popular applications in control systems, where an FIS is built up by codifying human knowledge as linguistic IF-THEN rules. Since its first reported industrial application in 1982 [41], it has aroused global interest in the industrial and scientific community, and fuzzy logic has also been widely applied in data analysis, regression and prediction, as well as signal and image processing. Many application-specific integrated circuits (ASICs) has also been designed for fuzzy logic [31].

In this paper, we give a systematic introduction to fuzzy logic and neuro-fuzzy systems. The paper is organized as follows. In Section 2, we provide a short tutorial on fuzzy logic. Section 3 compares fuzzy logic and neural network paradigms. Section 4 compares the relation between fuzzy logic and MLP/RBFN, and rule generation from trained neural networks is introduced in this section. Rule extraction from numerical data is introduced in Section 5. The paradigm of neuro-fuzzy systems is described in Section 6. Some neuro-fuzzy models are introduced in Section 7. In Section 8, we describe some fuzzy neural circuits. An illustration of using neuro-fuzzy systems is given in Section 9. We summarize this paper in Section 10.

2. FUNDAMENTALS OF FUZZY LOGIC

2.1 Definitions

We list below some definitions and terminologies used in the fuzzy logic literature.

2.1.1 Universe of Discourse

The universal set X: $X \rightarrow [0,1]$ is called the universe of discourse, or simply the universe. The implication $X \rightarrow [0,1]$ is the abbreviation for the IF-THEN rule: "IF x is in X, THEN its MF $\mu_X(x)$ is in [0,1].", where $\mu_X(x)$ is the MF of x. The universe X may contain either discrete or continuous values.

2.1.2 Linguistic Variable

A linguistic variable is a variable whose values are linguistic terms in a natural or artificial language. For example, the size of an object is a linguistic variable, whose value can be *small*, *medium*, and *big*.

2.1.3 Fuzzy Set

A fuzzy set A in X is defined by

$$A = \{x, \mu_A(x) | x \in X\}, \tag{1}$$

where $\mu_A(x) \in [0,1]$ is the MF of x in A. For $\mu_A(x)$, the value 1 stands for complete membership of the set A, while 0 represents that x does not belong to the set at all. A fuzzy set can also be syntactically represented by

$$A = \begin{cases} \sum_{x_i \in X} \frac{\mu_A(x_i)}{x_i}, & \text{if } X \text{ is discrete} \\ \int_X \frac{\mu_A(x)}{x}, & \text{if } X \text{ is continuous} \end{cases}. \quad (2)$$

2.1.4 Support

The elements on fuzzy set A whose membership is larger than zero are called the support of A

$$\text{sp}(A) = \{x \in A | \mu_A(x) > 0\}. \quad (3)$$

2.1.5 Height

The height of a fuzzy set A is defined by

$$\text{hgt}(A) = \sup\{\mu_A(x) | x \in X\}. \quad (4)$$

2.1.6 Normal Fuzzy Set and Non-normal Fuzzy Set

A fuzzy set A is said to be *normal* if $\text{hgt}(A) = 1$. If $0 < \text{hgt}(A) < 1$, the fuzzy set A is said to be *non-normal*. The non-normal fuzzy set can be normalized by dividing the height of A, i.e., $\bar{\mu}_A(x) = \frac{\mu_A(x)}{\text{hgt}(A)}$.

2.1.7 Fuzzy Subset

A fuzzy set $A = \{(x, \mu_A(x)) | x \in X\}$ is said to be a fuzzy subset of $B = \{(x, \mu_B(x)) | x \in X\}$ if $\mu_A(x) \leq \mu_B(x)$, denoted by $A \subseteq B$.

2.1.8 Fuzzy Partition

For a linguistic variable, a number of fuzzy subsets are enumerated as the value of the variable. This collection of fuzzy subsets is called a *fuzzy partition*. Each fuzzy subset has a MF. For a finite fuzzy partition $\{A_1, A_2, \cdots, A_n\}$ of a set A, the MF for each $x \in A$ satisfies

$$\sum_{i=1}^{n} \mu_{A_i}(x) = 1, \quad (5)$$

and A_i is normal. A fuzzy partition is illustrated in Fig. 1.

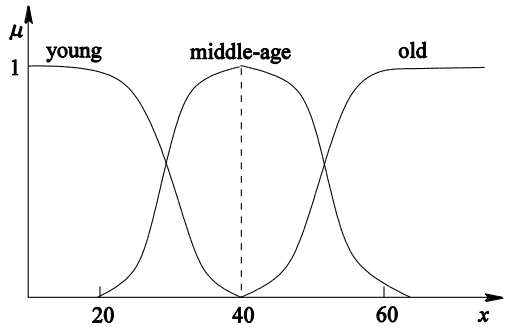


FIGURE 1: A fuzzy partition of human age. The fuzzy set for representing the linguistic variable *human age* is partitioned into three fuzzy subsets, namely, *young*, *middle-age*, *old*.
Each fuzzy subset is characterized by an MF.

2.1.9 Empty Set

The subset of X having no element is called the *empty set*, denoted by $\emptyset$.

2.1.10 Complement

The complement of A, written $\bar{A}$, $\neg A$ or NOT A, is defined as $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$. Thus, $\bar{X} = \emptyset$ and $\bar{\emptyset} = X$.

2.1.11 α-cut

The α-*cut* or α-*level set* of a fuzzy set A, written $\mu_A[\alpha]$, is defined as

$$\mu_A[\alpha] = \{x \in A | \mu_A(x) \geq \alpha\}, \tag{6}$$

where $\alpha \in [0,1]$. For continuous sets, $\mu_A\,[\alpha]$ can be characterized by an interval or a union of intervals.

2.1.12 Kernel or Core

All the elements in a fuzzy set A with membership degree 1 constitute a subset called the kernel or core of the fuzzy set, written as $\mathrm{co}(A) = \mu_A\,[1]$.

2.1.13 Convex Fuzzy Set

A fuzzy set A is said to be convex if and only if

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \mu_A(x_1) \wedge \mu_A(x_2) \tag{7}$$

for $\lambda \in [0,1]$, and $x_1, x_2 \in X$, where $\wedge$ denotes the minimum operation. Any α-cut set of a convex fuzzy set is a closed interval.

2.1.14 Concave Fuzzy Set

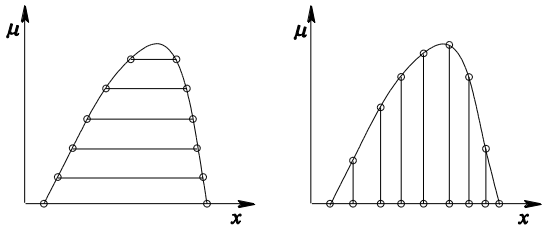
A fuzzy set A is said to be concave if and only if

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \leq \mu_A(x_1) \vee \mu_A(x_2). \tag{8}$$

For $\lambda \in [0,1]$, and $x_1, x_2 \in X$, where $\vee$ denotes the maximum operation.

2.1.15 Fuzzy Number

A fuzzy number A is a fuzzy set of the real line with a normal, convex and continuous MF of bounded support. A fuzzy number is usually represented by a family of α-level sets or by a discretized MF, as illustrated in Fig. 2.



(a) (b)

FIGURE 2: Representations of a fuzzy number. (a) α-level sets. (b) Discretized MF.

2.1.16 Fuzzy Singleton

A fuzzy set $A = \{(x, \mu_A(x)) | x \in X\}$ is said to be a *fuzzy singleton* if $\mu_A(x) = 1$ for $x \in X$ and $\mu_A(\mathrm{x}') = 0$ for all $\mathrm{x}' \in X$ with $\mathrm{x}' \neq x$.

2.1.17 Hedge

A hedge transforms a fuzzy set into a new fuzzy set. A hedge operator is comparable to an adverb in English. Hedges are used to intensify or dilute the characteristic of a fuzzy set such as *very* and *quite*, or to approximate a fuzzy set or convert a scalar to a fuzzy set such as *roughly*. For example, for a fuzzy set *strong* with membership degree $\mu_A(x)$, *very strong* can be described using the membership degree $\mu_A^2(x)$, while *quite strong* can be described using the membership degree $\mu_A^{\frac{1}{2}}(x)$.

2.1.18 Extension Principle

Given mapping $f: X \to Y$ and a fuzzy set $A = \{ (x, \mu_A(x)) | x \in X \}$, the extension principle is given by

$$f(A) = \{ (f(x), \mu_A(x)) | x \in X \}. \quad (9)$$

2.1.19 Cartesian Product

If X and Y are two universal sets, then $X \times Y$ is the set of all ordered pairs (x, y) for $x \in X$ and $y \in Y$. Let A be a fuzzy set of X and B a fuzzy set of Y. The Cartesian product is defined as

$$A \times B = \{ (z, \mu_{A \times B}(z)) | z = (x, y) \in Z, Z = X \times Y \}, \quad (10)$$

where $\mu_{A \times B}(z) = \mu_A(x) \wedge \mu_B(x)$, $\wedge$ denoting the t-norm operation.

2.1.20 Fuzzy Relation

Fuzzy relation is used to describe the association between two things. If R is a subset of $X \times Y$, then R is said to be a relation between X and Y, or a relation on $X \times Y$. Mathematically,

$$R(x, y) = \{ ((x, y), \mu_R(x, y)) | (x, y) \in X \times Y, \mu_R(x, y) \in [0,1] \}, \quad (11)$$

where $\mu_R(x, y)$ is the degree of membership for association between x and y. A fuzzy relation is also a fuzzy set.

2.1.21 Fuzzy Matrix and Fuzzy Graph

Given finite, discrete fuzzy sets $X = \{ x_1, x_2, \cdots, x_m \}$ and $Y = \{ y_1, \ldots, y_n \}$, a fuzzy relation on $X \times Y$ can be represented by an $m \times n$ matrix $\mathbf{R} = [R_{ij}] = [\mu_R (x_i, y_j)]$. This matrix is called a *fuzzy matrix*. The fuzzy relation R can be represented by a fuzzy graph. In a fuzzy graph, all x_i and y_j are vertices, and the grade $\mu_R(x_i, y_j)$ is added to the connection from x_i and y_j.

2.1.22 *t*-norm

A mapping $T: [0,1] \times [0,1] \to [0,1]$ with the following four properties is called *t-norm*. For all $x, y, z \in [0,1]$,

- Commutativity: $T(x, y) = T(y, x)$;
- Monotonicity: $T(x, y) \leq T(x, z)$, if $y \leq z$;
- Associativity: $T(x, T(y, z)) = T(T(x, y), z)$;
- Linearity: $T(x, 1) = x$.

2.1.23 *t*-conorm

A mapping $C: [0,1] \times [0,1] \to [0,1]$ having the following four properties is called *t-conorm*. For all $x, y, z \in [0,1]$,

- Commutativity: $C(x, y) = C(y, x)$;
- Monotonicity: $C(x, y) \leq C(x, z)$, if $y \leq z$;

- Associativity: $C(x, C(y, z)) = C(C(x, y), z)$;
- Linearity: $C(x, 0) = x$.

2.2 Membership Function

A fuzzy set A over the universe of discourse X, $A \subseteq X \to [0,1]$, is described by the degree of membership $\mu_A(x) \in [0,1]$ for each $x \in X$. Unimodality and normality are two important aspects of the MFs [23]. Piecewise-linear functions such as triangles and trapezoids are popular MFs. The triangular MFs can be defined by

$$\mu(x; a, b, c) = \begin{cases} \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b < x \le c \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where the shape parameters satisfies $a \le b \le c$, and $b \in X$. Triangular MFs are useful for modeling fuzzy numbers or linguistic terms such as "The temperature is about $20°$C". The trapezoid MFs have flat top with constant value 1. Trapezoid MFs are suitable for modeling such linguistic terms as "He looks like a teenager".

The Gaussian and bell-shaped functions have continuous derivatives, and are usually used to replace the triangular MF when shape parameters are adapted using the gradient-descent procedure. Another popular MF is a sigmoidal functions of the form

$$\mu(x; c, \beta) = \frac{1}{1 + \mathrm{e}^{-\beta(x-c)}}, \tag{13}$$

where c shifts the function to the left or to the right, and β controls the shape of the function. When $\beta > 1$ it is an S-shaped function, and when $\beta < -1$ it is a Z-shaped function. By multiplying an S-shaped function by a Z-shaped function, a π-shaped function is obtained [29]. π-shaped MFs can be used in situations similar to that where trapezoid MFs are used.

2.3 Intersection and Union

The set operations intersection and union correspond to the logic operations conjunction (*AND*) and disjunction (*OR*), respectively. Intersection is described by the so-called triangular norm (t-norm), denoted by $T(x, y)$, whereas union is described by the so-called triangular conorm (t-conorm), denoted by $C(x, y)$.

If A and B are fuzzy subsets of X, then intersection $I = A \cap B$ is defined by

$$\mu_I(x) = T\big(\mu_A(x), \mu_B(x)\big). \tag{14}$$

Basic t-norms are given as the standard intersection, the bound sum, the algebraic product and the drastic intersection [14]. The popular standard intersection and algebraic product are respectively defined by

$$T_{\mathrm{m}}(x, y) = \min(x, y), \tag{15}$$

$$T_{\mathrm{p}}(x, y) = xy. \tag{16}$$

Similarly, union $U = A \cup B$ is defined by

$$\mu_U(x) = C\big(\mu_A(x), \mu_B(x)\big). \tag{17}$$

The corresponding basic t-conorms are given as the standard union, the bounded sum, the algebraic sum, and the drastic union [14]. Corresponding to the standard intersection and algebraic product, the two popular t-conorms are respectively the standard union and the algebraic sum

$$C_{\mathrm{m}}(x,y) = \max(x,y), \tag{18}$$

$$C_{\mathrm{p}}(x,y) = x + y - xy. \tag{19}$$

When the t-norm and the t-conorm satisfy $1 - T(x,y) = C(1-x, 1-y)$, they are said to be *dual*. This makes De Morgan's laws $\overline{\mathrm{A} \cap \mathrm{B}} = \overline{\mathrm{A}} \cup \overline{\mathrm{B}}$ and $\overline{\mathrm{A} \cup \mathrm{B}} = \overline{\mathrm{A}} \cap \overline{\mathrm{B}}$ to still hold in fuzzy set theory. The above t-norms and t-conorms with the same subscripts are dual. To satisfy the principle of duality, they are usually used in pairs.

2.4 Aggregation, Fuzzy Implication, and Fuzzy Reasoning

Aggregation or composition operations on fuzzy sets provide a means for combining several sets in order to produce a single fuzzy set. T-conorms are usually used as aggregation operators. Consider the relations

$$R_1(x,y) = \left\{ \left((x,y), \mu_{R_1}(x,y) \right) \middle| (x,y) \in X \times Y, \mu_{R_1}(x,y) \in [0,1] \right\}, \tag{20}$$

$$R_2(y,z) = \left\{ \left((y,z), \mu_{R_2}(y,z) \right) \middle| (y,z) \in Y \times Z, \mu_{R_2}(y,z) \in [0,1] \right\}. \tag{21}$$

The max-min composition, denoted by $R_1 \circ R_2$ with MF $\mu_{R_1 \circ R_2}$, is defined by

$$R_1 \circ R_2 = \left\{ \left((x,z), \max_y \left\{ \min\left(\mu_{R_1}(x,y), \mu_{R_2}(y,z) \right) \right\} \right) \middle| (x,z) \in X \times Z, y \in Y \right\}. \tag{22}$$

There are some other composition operations, such as the min-max composition, denoted by $R_1 \diamond R_2$ with the difference that the role of max and min are interchanged. The two compositions are related by $\overline{R_1 \diamond R_2} = \overline{R_1} \circ \overline{R_2}$.

Fuzzy implication is used to represent fuzzy rules. It is a mapping $f: A \to B$ according to the fuzzy relation R on $A \times B$

$$\left(y, \mu_B(y)\right) = f\left(\left(x, \mu_A(x)\right) \right). \tag{23}$$

Denote p as "x is A" and q as "y is B", then (23) can be stated as $p \to q$ (if p then q). For a fuzzy rule expressed as a fuzzy implication using the defined fuzzy relation R, the output linguistic variable B is denoted by $B = A \circ R$, which is characterized by $\mu_B(y) = \vee_x \left(\mu_A(x) \wedge \mu_R(x,y) \right)$.

Fuzzy reasoning, also called approximate reasoning, is an inference procedure for deriving conclusions from a set of fuzzy rules and one or more conditions [51]. The compositional rule of inference is the essential rational behind fuzzy reasoning. A simple example of fuzzy reasoning is described here. Consider the fuzzy set $A = \left\{ \left(x, \mu_A(x) \right) \middle| x \in X \right\}$ and the fuzzy relation R on $A \times B$, given by $R(x,y) = \left\{ \left((x,y), \mu_R(x,y) \right) \middle| (x,y) \in X \times Y \right\}$. Fuzzy set B can be inferred from fuzzy set A and their fuzzy relation $R(x,y)$ by the max-min composition

$$B = A \circ R = \left\{ \left(y, \max_x \left\{ \min\left(\mu_A(x), \mu_R(x,y) \right) \right\} \right) \middle| x \in X, y \in Y \right\}. \tag{24}$$

2.5 Fuzzy Inference Systems

In control systems, the inputs to the systems are the error and the change in the error of the feedback loop, while the output is the control action. Fuzzy logic-based controllers are popular control systems. The general architecture of a fuzzy controller is depicted in Fig. 3. The core of a

fuzzy controller is an FIS, in which the data flow involves fuzzification, knowledge base evaluation, and defuzzification. In an FIS, sometimes termed a fuzzy system or a fuzzy model, the knowledge base is comprised of the fuzzy rule base and the database. The database contains the linguistic term sets considered in the linguistic rules and the MFs defining the semantics of the linguistic variables, and information about domains. The rule base contains a collection of linguistic rules that are joined by the ALSO operator. Expert provides his knowledge in the form of linguistic rules. The fuzzification process collects the inputs and then converts them into linguistic values or fuzzy sets. The decision logic, called fuzzy inference engine, generates output from the input, and finally the defuzzification process produces a crisp output for control action.

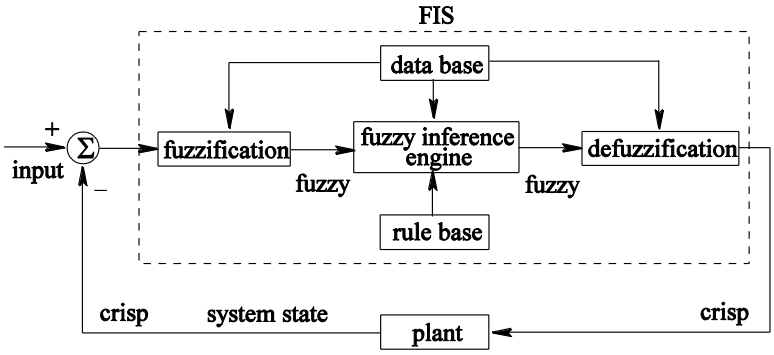


FIGURE 3: The architecture of a fuzzy controller. The core of the fuzzy controller is an FIS.

Interpretations of a certain rule or the rule base depends on the FIS model. The Mamdani [69] and the TSK [103] models are two popular FISs. The Mamdani model is a nonadditive fuzzy model that aggregates the output of fuzzy rules using the maximum operator, while the TSK model is an additive fuzzy model that aggregates the output of rules using the addition operator. Kosko's standard additive model (SAM) [56] is another additive fuzzy model. All these models can be derived from fuzzy graph [122], and are universal approximators [55], [113], [13], [15], [75]. When approximating an unknown control function, neural networks achieve a solution using the learning process, while FISs apply a vague interpolation technique. Unlike neural networks and other numerical models, fuzzy models operate at a level of information granules—fuzzy sets.

2.6 Fuzzy Rules and Fuzzy Interference

Fuzzy mapping rules and fuzzy implication rules are the two types of fuzzy rules [122]. A fuzzy mapping rule describes a functional mapping relationship between inputs and an output using linguistic terms, while a fuzzy implication rule describes a generalized logic implication relationship between two logic formulas involving linguistic variables. Fuzzy implication rules generalize set-to-set implications, whereas fuzzy mapping rules generalize set-to-set associations. The former was motivated to allow intelligent systems to draw plausible conclusions in a way similar to human reasoning, while the latter was motivated to approximate complex relationships such as nonlinear functions in a cost-effective and easily comprehensible way. The foundation of fuzzy mapping rule is fuzzy graph, while the foundation of fuzzy implication rule is a generalization to two-valued logic.

A rule base consists of a number of rules given in the form "IF $condition$, THEN $action$". The condition, also called premise, is made up of a number of antecedents that are negated or combined by different operators such as AND or OR computed with t-norms or t-conorms. In a fuzzy rule system, MFs for fuzzy subsets can be selected according to human intuition, or by learning from training data.

A fuzzy inference is made up of several rules with the same output variables. Given a set of fuzzy rules, the inference result is a combination of the fuzzy values of the conditions and the corresponding actions. For example, we have a set of N_{r} rules

$$\mathrm{R}_i\text{: IF } (condition = C_i) \text{ THEN } (action = A_i)$$

for $i = 1, \dots, N_{\mathrm{r}}$, where C_i is a fuzzy set. Assuming that a condition has a membership degree of μ_i associated with the set C_i. The condition is first converted into a fuzzy category using a syntactical representation, $condition = \sum_i^{N_{\mathrm{r}}} \frac{C_i}{\mu_i}$. We can see each rule is valid to a certain extent. A fuzzy inference is the combination of all the possible consequences. The action coming from a fuzzy inference is also a fuzzy category, with a syntactical representation

$$action = \frac{A_1}{\mu_1} + \frac{A_2}{\mu_2} + \cdots + \frac{A_{N_{\mathrm{r}}}}{\mu_{N_{\mathrm{r}}}}. \tag{25}$$

The inference procedure depends on fuzzy reasoning. This result can be further processed or transformed into a crisp value.

2.7 Fuzzification and Defuzzification

Fuzzification is to transform crisp inputs into fuzzy subsets. Given crisp inputs x_i, $i = 1, \dots, n$, fuzzification is to construct the same number of fuzzy sets A^i,

$$A^i = \mathrm{fuzz}(x_i), \tag{26}$$

where $\mathrm{fuzz}(\cdot)$ is a fuzzification operator. Fuzzification is determined according to the defined MFs.

Defuzzification is to map fuzzy subsets of real numbers into real numbers. In an FIS, defuzzification is applied after aggregation. Popular defuzzification methods include the centroid defuzzifier [69], and the mean-of-maxima defuzzifier [69]. The centroid defuzzifier is the best-known method, which is to find the centroid of the area surrounded by the MF and the horizontal axis [52]. Aggregation and defuzzification can be combined into a single phase, such as the weighted-mean method [36]

$$\mathrm{defuzz}(B) = \frac{\sum_{i=1}^{N_{\mathrm{r}}} \mu_i b_i}{\sum_{i=1}^{N_{\mathrm{r}}} \mu_i}, \tag{27}$$

where N_{r} is the number of rules, μ_i is the degree of activation of the ith rule, and b_i is a numeric value associated with the consequent of the ith rule, B_i. The parameter b_i can be selected as the mean value of the α-level set when α is equal to μ_i [36].

2.8 Mamdani Model

Given a set of N examples $\{(\mathbf{x}_p, \mathbf{y}_p) | \mathbf{x}_p \in R^n, \mathbf{y}_p \in R^m\}$, the underlying system can be identified by using the Mamdani or the TSK model.

For the Mamdani model with N_{r} rules, the ith rule is given by

$$\mathrm{R}_i\text{: IF } \mathbf{x} \text{ is } A_i, \text{ THEN } \mathbf{y} \text{ is } B_i$$

for $i = 1, \dots, N_{\mathrm{r}}$, where $A_i = \{A_i^1, A_i^2, \dots, A_i^n\}$, $B_i = \{B_i^1, B_i^2, \dots, B_i^m\}$, and A_i^j and B_i^k are respectively fuzzy sets that define an input and output space partitioning.

For an n-tuple input in the form of "$\mathbf{x}$ is A'", the system output "$\mathbf{y}$ is B'" is characterized by combining the rules according to

$$\mu_{B'}(\mathbf{y}) = \bigvee_{i=1}^{N_\mathrm{r}} \left(\mu_{A_i'}(\mathbf{x}) \wedge \mu_{B_i}(\mathbf{y}) \right), \tag{28}$$

where the fuzzy partitioning $A' = \{A'^1, A'^2, \ldots, A'^n\}$ and $B' = \{B'^1, B'^2, \ldots, B'^m\}$,

$$\mu_{A_i'}(\mathbf{x}) = \mu_{A'}(\mathbf{x}) \wedge \mu_{A_i}(\mathbf{x}) = \bigwedge_{i=1}^{n} \left(\mu_{A'^j} \wedge \mu_{A_i^j} \right). \tag{29}$$

$\mu_{A'}(\mathbf{x}) = \wedge_{j=1}^{n} \mu_{A'^j}$ and $\mu_{A_i}(\mathbf{x}) = \wedge_{j=1}^{n} \mu_{A_i^j}$ being respectively the membership degrees of $\mathbf{x}$ to the fuzzy sets A' and A_i, $\mu_{B_i}(\mathbf{y}) = \wedge_{k=1}^{m} \mu_{B_i^k}$ is the membership degree of $\mathbf{y}$ to the fuzzy set B_i, $\mu_{A'^j_i}$ is the association between the jth input of A' and the ith rule, $\mu_{B_i^k}$ is the association between the kth input of B and the ith rule, $\wedge$ is the intersection operator, and $\vee$ is the union operator.

When minimum and maximum are respectively used as the intersection and union operators, the Mamdani model is called a max-min model. We now illustrate the inference procedure for the Mamdani model. Assume that we have a two-rule Mamdani FIS with the rules of the form

R_i: IF x_1 is A_i and x_2 is B_i, THEN y is C_i

for $i = 1,2$. When the max-min composition is employed, for the inputs "x_1 is A'" and "x_2 is B'", the fuzzy reasoning procedure for the output y is illustrated in Fig. 4. A defuzzification strategy is needed to get crisp output value.

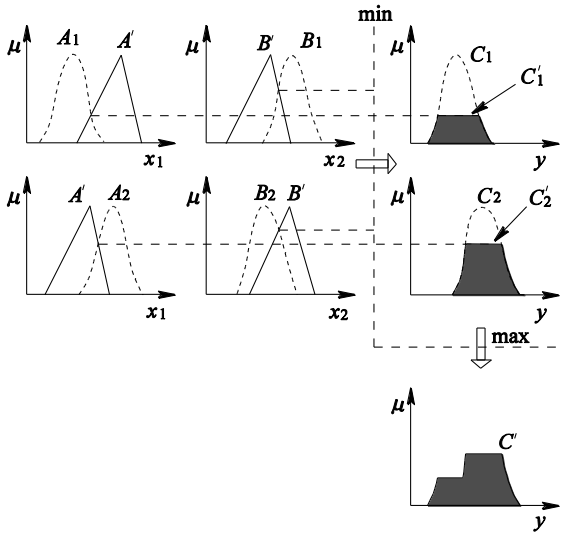


FIGURE 4: The inference procedure of the Mamdani model with the min and max operators and fuzzy inputs.

The Mamdani model offers a high semantic level and a good generalization capability. It contains fuzzy rules built from expert knowledge. However, FISs based only on expert knowledge may result in insufficient accuracy. For accurate numerical approximation, the TSK model can usually generate a better performance.

2.9 Takagi-Sugeno-Kang Model

In the TSK model [103], for the same set of examples $\{(\mathbf{x}_\mathrm{p}, \mathbf{y}_\mathrm{p})\}$, fuzzy rules are given in the form

$$\mathrm{R}_i\text{: IF } \mathbf{x} \text{ is } A_i\text{, THEN } \mathbf{y} = \mathbf{f}_i(\mathbf{x})$$

for $i = 1,2,\dots,N_\mathrm{r}$, where $\mathbf{f}_i(\mathbf{x}) = \left(f_i^1(\mathbf{x}),\dots,f_i^m(\mathbf{x})\right)^T$ is a crisp vector function of $\mathbf{x}$; usually $f_i^j(\mathbf{x})$ is selected as a linear relation with $f_i^j(\mathbf{x}) = \left(\mathbf{a}_i^j\right)^T\mathbf{x} + b_i^j$, where $\mathbf{a}_i^j$ and b_i^j are adjustable parameters.

For an n-tuple input in the form of "$\mathbf{x}$ is A'", the output $\mathbf{y}'$ is obtained by combining the rules according to

$$\mathbf{y}' = \frac{\sum_{i=1}^{N_\mathrm{r}} \mu_{A_i'}(\mathbf{x})\mathbf{f}_i(\mathbf{x})}{\sum_{i=1}^{N_\mathrm{r}} \mu_{A_i'}(\mathbf{x})}, \tag{30}$$

where $\mu_{A_i'}(\mathbf{x})$ is defined by (29), and can be derived by the procedure shown in the left part of Fig. 4. This model produces a real-valued function, and it is essentially a model-based fuzzy control method. The stability analysis of the TSK model is given in [104]. The TSK model typically selects $f_i^j(\cdot)$ as first-order polynomials, hence the model termed the first-order TSK model. When $f_i^j(\cdot)$ are selected as constants, it is called the zero-order TSK model and can be regarded as a special case of the Mamdani model.

In comparison with the Mamdani model, the TSK model, which is based on automatic learning from the data, can accurately approximate a function using fewer rules. It has a stronger and more flexible representation capability than the Mamdani mode. In the TSK model, rules are extracted from the data, but the generated rules may have no meaning for experts. The TSK model has found more successful applications in building fuzzy systems.

2.10 Complex Fuzzy Logic

Complex fuzzy sets and logic are mathematical extensions of fuzzy sets and logic from the real domain to the complex domain [87], [86]. A complex fuzzy set S is characterized by a complex-valued MF, and membership of any element x in S is given by a complex-valued membership degree of the form

$$\mu_S(x) = r_S(x)\mathrm{e}^{\mathrm{j}\varphi_S(x)}, \tag{31}$$

where the amplitude $r_S(x) \in [0,1]$, and φ_S is the phase. Thus, $\mu_S(x)$ is within a unit circle in the complex plane.

In [87], [86], basic set operators for fuzzy logic have been extended for the complex fuzzy logic, and some additional operators such as the vector aggregation, set rotation and set reflection, are also defined. The operations of intersection, union and complement for complex fuzzy sets are defined only on the modulus of the complex membership degree. In [27], the complex fuzzy logic is extended to a logic of vectors in the plane, rather than scalar quantities. In [74], a complex fuzzy set is defined as an MF mapping the complex plane into $[0,1] \times [0,1]$.

Complex fuzzy sets are superior to the Cartesian products of two fuzzy sets. Complex fuzzy logic maintains both the advantages of the fuzzy logic and the properties of complex fuzzy sets. In complex fuzzy logic, rules constructed are strongly related and a relation manifested in the phase term is associated with complex fuzzy implications. In a complex FIS, the output of each rule is a complex fuzzy set, and phase terms are necessary when combining multiple rules so as to generate the final output. Complex FISs are useful for solving some hard problems for traditional fuzzy methods, in which rules are related to one another with the nature of the relation varying as a function of the input to the system [86].

The fuzzy complex number [11], introduced by incorporating the complex number into the support of the fuzzy set, is a different concept from the complex fuzzy set [87]. A fuzzy complex number is

a fuzzy set of complex numbers, which have real-valued membership degree in the range $[0,1]$. An α-cut of a fuzzy complex number is based on the modulus of the complex numbers in the fuzzy set. A fuzzy complex number is a fuzzy set in one dimension, while a complex fuzzy set or number is a fuzzy set in two dimensions.

3. FUZZY LOGIC VS. NEURAL NETWORKS

Like FNNs, many fuzzy systems are proved to be universal approximators [63], [50], [13], [35], [57], [118]. In [63], the Mamdani model and FNNs are shown to be able to approximate each other to an arbitrary accuracy. The equivalence between the TSK model and the RBFN under certain conditions has been established in [50], [43] and the equivalence between fuzzy expert systems and neural networks has been proved in [13]. Gaussian-based Mamdani systems have the ability of approximating any sufficiently smooth function and reproducing its derivatives up to any order [35]. In [57], fuzzy systems with Gaussian MFs have been proved to be universal approximators for a smooth function and its derivatives.

From the viewpoint of an expert system, fuzzy systems and neural networks are quite similar as inference systems. An inference system involves knowledge representation, reasoning, and knowledge acquisition: (1) A trained neural network represents knowledge using connection weights and neurons in a distributed manner, while in a fuzzy system knowledge is represented using IF-THEN rules; (2) For each input, the trained neural network generates an output and this pure numerical procedure can be treated as a reasoning process, while reasoning in a fuzzy system is logic-based; (3) Knowledge acquisition is via learning in a neural network, while for a fuzzy system knowledge is encoded by a human expert. Both neural networks and fuzzy systems are dynamic, parallel distributed processing systems that estimate functions without any mathematical model and learn from experience with sample data.

Fuzzy systems can be applied to problems with knowledge represented in the form of IF-THEN rules. Problem-specific a priori knowledge can be integrated into the systems. Training pattern set and system modeling are not needed, and only heuristics are used. During the tuning process, one needs to add, remove, or change a rule, or even change the weight of a rule. This process, however, requires the knowledge of experts. On the other hand, neural networks are useful when we have training pattern set. We do not need any knowledge of the modeling of the problem. A trained neural network is a black box that represents knowledge in its distributed structure. However, any prior knowledge of the problem cannot be incorporated into the learning process. It is difficult for human beings to understand the internal logic of the system. Nevertheless, by extracting rules from neural networks, users can understand what neural networks have learned and how neural networks predict.

4. FUZZY INFERENCE SYSTEMS AND NEURAL NETWORKS

4.1 Fuzzy Inference Systems and Multilayer Perceptrons

For a three-layer (J_1-J_2-J_3) MLP, if the activation function in the hidden layer $\phi^{(1)}(\cdot)$ is selected as the logistic function $\phi^{(1)}(x) = \frac{1}{1+e^{-x}}$ and the activation function in the output layer $\phi^{(2)}(\cdot)$ is selected as the linear function $\phi^{(2)}(x) = x$, there always exists a fuzzy additive system that calculates the same function as the network does [7]. In [7], a fuzzy logic operator, called interactive-or (i-or), is defined by applying the concept of f-duality to the logistic function. The use of the i-or operator explains clearly the acquired knowledge of a trained MLP. The i-or operator is defined by [7]

$$a \otimes b = \frac{a \cdot b}{(1-a) \cdot (1-b) + a \cdot b}. \quad (32)$$

The i-or operator works on $(0,1)$. It is a hybrid between both a t-norm and a t-conorm. Based on the i-or operator, the equality between MLPs and FISs is thus established [7]. The equality proof also yields an automated procedure for knowledge acquisition. An extension of the method has been presented in [16], where the fuzzy rules obtained are in agreement with the domain of the input variables and a new logical operator, similar to, but with a higher representational power than the i-or, is defined.

In [32], relations between input uncertainties and fuzzy rules have been established. Sets of crisp logic rules applied to uncertain inputs are shown to be equivalent to fuzzy rules with sigmoidal MFs applied to crisp inputs. Integration of a reasonable uncertainty distribution for a fixed rule threshold or interval gives a sigmoidal MF. Crisp logic and fuzzy rule systems are shown to be respectively equivalent to the logical network and the three-layer MLP. Keeping fuzziness on the input side enables easier understanding of the networks or the rule systems. In [17], [100], MLPs are interpreted by fuzzy rules in such a way that the sigmoidal activation function is decomposed into three partitions, and represented by three TSK fuzzy rules with one TSK fuzzy rule for each partition. Each partition has its own MF. Accordingly, the value of the activation function at a point can be derived by the TSK model.

A fuzzy set is usually represented by a finite number of its supports. In comparison with conventional MF based FISs, α-cut based FISs [109] have a number of advantages. They can considerably reduce the required memory and time complexity, since they depend on the number of membership-grade levels, and not on the number of elements in the universes of discourse. Secondly, the inference operations can be performed for each α-cut set independently, and this enables parallel implementation. An α-cut based FIS can also easily interface with two-valued logic since the α-level sets themselves are crisp sets. In addition, fuzzy set operations based on the extension principle can be performed efficiently using α-level sets [109], [64]. For α-cut based FISs, each fuzzy rules can be represented as a pattern pair of degrees of membership at those points of the MFs obtained by dividing the intervals of the fuzzy sets linearly or by α-cut can be implemented by an MLP with the backpropagation (BP) rule. This is a learning problem of N_{r} samples with n inputs and m outputs.

4.2 Fuzzy Inference Systems and Radial Basis Function Networks

When the t-norm in the TSK model is selected as multiplication and the MFs are selected the same as RBFs in the normalized RBFN model, the two models are mathematically equivalent [50], [48]. Note that each hidden unit corresponds to a fuzzy rule. Normalized RBFNs provide a localized solution that is amenable to rule extraction. The receptive fields of some RBFs should overlap to prevent incompleteness of fuzzy partitions. To have a perfect match between the RBFs $\phi\left(\|\mathbf{x}-\mathbf{c}_i\|\right)$ and $\mu_{A_i'}(\mathbf{x})$ in (30), $\phi\left(\|\mathbf{x}-\mathbf{c}_i\|\right)$ should be factorizable in each dimension such that each component $\phi\left(|x_j - c_{i,j}|\right)$ corresponds to an MF $\mu_{\boldsymbol{A'^j}}$. The Gaussian RBF is the only strictly factorizable function.

In the normalized RBFN, w_{ij} 's typically take constant values and the normalized RBFN corresponds to the zero-order TSK model. When the RBF weights are linear regression functions of the input variables [59], [91], the model is functionally equivalent to the first-order TSK model.

When implementing the TSK model, one can select some $\mu_{A'^j_i} = 1$ or some $\mu_{A'^j_i} = \mu_{A'^j_k}$ in order to

increase the distinguishability of the fuzzy partitions. Correspondingly, one should share some component RBFs or set some component RBFs to unity [52]. This considerably reduces the effective number of free parameters in the RBFN. A distance measure like the Euclidean distance is used to describe the similarity between two component RBFs. After applying a clustering technique to locate prototypes and adding a regularization term describing the total similarity between all the RBFs and the shared RBF to the MSE function, a gradient-descent procedure is conducted so as to extract interpretable fuzzy rules from a trained RBFN [52]. The method can be applied to RBFNs with constant or linear regression weights. A fuzzy system can be first constructed according to heuristic knowledge and existing data, and then converted into an RBFN. This is followed by a refinement of the RBFN using a learning algorithm. Due to this learning procedure, the interpretability of the original fuzzy system may be lost. The RBFN is then again converted into interpretable fuzzy system, and knowledge is extracted from the network. This process refines the original fuzzy system design. The algorithm for rule extraction from the RBFN is given in [52].

In [107], normalized Gaussian RBFNs can be generated from simple probabilistic rules and probabilistic rules can also be extracted from trained RBFNs. Methods for reducing network complexity have been presented in order to obtain concise and meaningful rules. Two algorithms for rule extraction from RBFNs, which respectively generate a single rule describing each class and a single rule from each hidden unit, are given in [70]. Existing domain knowledge in rule format can be inserted into an RBFN as an initialization of optimal network training.

4.3 Rule Generation from Trained Neural Networks

In addition to rule generation from trained MLPs and RBFNs, rule generation can also be performed on other trained neural networks [46], [106]). Rule generation involves rule extraction and rule refinement. Rule extraction is to extract knowledge from trained neural networks, while rule refinement is to refine the rules that are extracted from neural networks and initialized with crude domain knowledge.

Recurrent neural networks (RNNs) have the ability to store information over indefinite periods of time, develop hidden states through learning, and thus conveniently represent recursive linguistic rules [72]. They are particularly well-suited for problem domains, where incomplete or contradictory prior knowledge is available. In such cases, knowledge revision or refinement is also possible. Discrete-time RNNs can correctly classify strings of a regular language [80]. Rules defining the learned grammar can be extracted in the form of deterministic finite-state automata (DFAs) by applying clustering algorithms [29] in the output space of neurons. Starting from an initial network state, the algorithm searches the equally partitioned output space of N state neurons in a breadth-first manner. A heuristic is used to choose among the consistent DFAs that model, which best approximates the learned regular grammar. The extracted rules demonstrate high accuracy and fidelity and the algorithm is portable. Based on [80], an augmented RNN that encodes fuzzy finite-state automata (FFAs) and recognizes a given fuzzy regular language with an arbitrary accuracy has been constructed in [81]. FFAs are transformed into equivalent DFAs by using an algorithm that computes fuzzy string membership. FFAs can model dynamical processes whose current state depends on the current input and previous states. The granularity within both extraction techniques is at the level of ensemble of neurons, and thus, the approaches are not strictly decompositional.

RNNs are suitable for crisp/fuzzy grammatical inference. A method that uses a SOM for extracting knowledge from an RNN [9] is able to infer a crisp/fuzzy regular language. Rule extraction is also carried out upon Kohonen networks [110]. A comprehensive survey on rule generation from trained neural networks is given from a softcomputing perspective in [72], where the optimization capability of evolutionary algorithms (EAs) are emphasized for rule refinement.

Rule extraction from RNNs aims to find models of an RNN, typically in the form of finite state machines. A recent overview of rule extraction from RNNs is given in [47].

4.4 Extracting Rules from Numerical Data

FISs can be designed directly from expert knowledge and data. The design process is usually decomposed into two phases, namely, rule generation and system optimization [39]. Rule generation leads to a basic system with a given space partitioning and the corresponding set of rules, while system optimization gives the optimal membership parameters and rule base. Design of fuzzy rules can be performed in one of three ways, namely, all the possible combinations of fuzzy partitions, one rule for each data pair, or dynamically choosing the number of fuzzy sets.

For good interpretability, a suitable selection of variables and the reduction of the rule base are necessary. During the system optimization phase, merging techniques such as cluster merging and fuzzy set merging are usually used for interpretability purposes. Fuzzy set merging leads to a higher interpretability than cluster merging. The reduction of a set of rules results in a loss of numerical performance on the training data set, but a more compact rule base has a better generalization capability and is also easier for human understanding. EAs [93] or learning [50] are also used for extracting fuzzy rules and optimizing MFs and rule base. Methods for designing FISs from data are analyzed and surveyed in [39]. They are grouped into several families and compared based on rule interpretability.

4.5 Rule Generation Based on Fuzzy Partitioning

Rule generation can be based on a partitioning of the multidimensional space. Fuzzy partitioning corresponds to structure identification for FISs, followed by parameter identification using a learning algorithm. There are usually three methods for partitioning the input space, namely, grid partitioning, tree partitioning, and scatter partitioning. These partitioning methods in the two-dimensional input space are illustrated in Fig. 5.

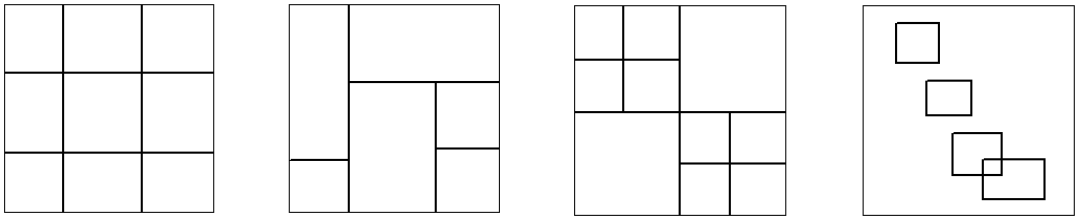


FIGURE 5: Partitioning of the two-dimensional input space. (a) Grid partitioning. (b) k-d tree partitioning. (c) Multilevel grid partitioning. (d) Scatter partitioning.

4.6 Grid Partitioning

The grid structure has easy interpretability and is most widely used for generating fuzzy rules. Fuzzy sets of each variable are shared by all the rules. However, the number of fuzzy rules grows exponentially with input dimension, namely, the curse-of-dimensionality problem. For n input variables, each being partitioned into m_i fuzzy sets, a total of $\prod_{i=1}^{n} m_i$ rules are needed to cover the whole input space. Since each rule has a few parameters to adjust, there are too many parameters to adapt during the learning process. Too many fuzzy rules also harm the interpretability of the fuzzy system. Thus, the method is appropriate for a small dimensional data set with a good coverage. A training procedure can be applied to optimize the grid structure and the rule consequences [50]. The grid structure is illustrated in Fig. 5 (a).

4.7 Tree Partitioning

k-d tree and multilevel grid structures are two hierarchical partitioning techniques that effectively relieve the problem of rule explosion [101]. The input space is first partitioned roughly, and a subspace is recursively divided until a desired approximation performance is achieved. The k-d tree results from a series of guillotine cuts. A guillotine cut is a cut that is entirely across the subspace to be partitioned. After the ith guillotine cut, the entire space is partitioned into $i + 1$ regions. Heuristics based on the distribution of training examples or parameter identification methods can usually be employed to find a proper k-d tree structure [101]. For the multilevel grid structure [101], the top-level grid coarsely partitions the whole space into equal-sized and evenly spaced fuzzy boxes, which are recursively partitioned into finer grids until a criterion is met. Hence, a multilevel grid structure is also called a box tree. The criterion can be that the resulting boxes have similar number of training examples or that an application-specific evaluation in each grid is below a threshold. A k-d tree partitioning and a multilevel grid partitioning are respectively illustrated in Fig. 5 (b) and (c). A multilevel grid in the two-dimensional space is called a quad tree. Tree partitioning needs some heuristics to extract rules and its application to high-dimensional problems faces practical difficulties.

4.8 Scatter Partitioning

Scatter partitioning usually generates fewer fuzzy regions than the grid and tree partitioning techniques owing to the natural clustering property of training patterns. Fuzzy clustering algorithms form a family of rule generation techniques. The training examples are gathered into homogeneous groups and a rule is associated to each group. The fuzzy sets are not shared by the rules, but each of them is tailored for one particular rule. Thus, the resulting fuzzy sets are usually difficult to interpret [39]. Clustering is well adapted for large work spaces with a small amount of training examples. However, scatter partitioning of high-dimensional feature spaces is difficult, and some learning or evolutionary procedures may be necessary. Clustering algorithms [29] can be applied for scatter partitioning. A scatter partitioning is illustrated in Fig. 5 (d). The curse of dimensionality can also be alleviated by reducing the input dimensions by discarding some irrelevant inputs or compressing the input space using feature selection or feature extraction techniques. Some clustering-based methods for extracting fuzzy rule for function approximation are proposed in [121], [20], [21], [4]. These methods are based on the TSK model. Clustering can be used for identification of the antecedent part of the model such as determination of the number of rules and initial rule parameters. The consequent part of the model can be estimated by the linear LS method. In [21], the combination of the subtractive clustering with the linear LS method provides an extremely fast and accurate method for fuzzy system identification, which is better than the adaptive-network-based FIS (ANFIS) [48]. Based on the Mamdani model, a clustering-based method for nonlinear regression is also given in [117].

4.9 Hierarchical Rule Generation

Hierarchical structure for fuzzy rule systems can also effectively solve the rule explosion problem [85], [114], [68]. A hierarchical fuzzy system is comprised of a number of low-dimensional fuzzy systems such as TSK systems connected in a hierarchical fashion. The total number of rules increases only linearly with the number of input variables. For example, for a hierarchical fuzzy system shown in Fig. 6, if there are n variables each of which is partitioned into m_i fuzzy subsets, the total number of rule is only $\sum_{i=1}^{n-1} m_i\, m_{i+1}$. Hierarchical TSK systems [114] and generalized hierarchical TSK systems [68] are universal approximators of any continuous function defined on a compact set.

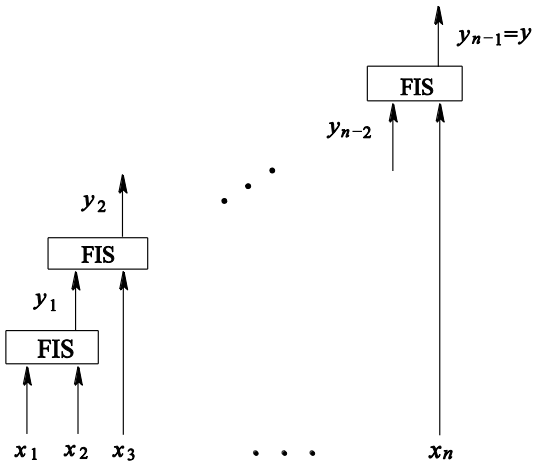


FIGURE 6: Example of a hierarchical fuzzy system with n inputs and one output. The system is comprised of $n-1$ two-input TSK systems. The n input variables are x_i, $i = 1, \ldots, n$, the output is denoted by y, and y_i is the output of the ith TSK system.

In Fig. 6, the n input variables are x_i, $i = 1, \ldots, n$, and the output is denoted by y. There exist relations

$$y_i = f_i(y_{i-1}, x_{i+1}) \tag{33}$$

for $i = 1, \ldots, n-1$, where f_i is the nonlinear relation described by the ith TSK system, y_i is the output of the ith TSK system, and $y_0 = x_1$. The final output is $y = y_{n-1}$. The output y is easily obtained by a recursive procedure. Thus, the inference in the hierarchical fuzzy system is in a recursive manner.

The hierarchical fuzzy system reduces the number of rules, however, the curse of dimensionality is inherent in the system. In the standard fuzzy system, the degree of freedom is unevenly distributed over the IF and THEN parts of the rules, with a comprehensive IF part to cover the whole domain and a simple THEN part. The hierarchical fuzzy system, on the other hand, provides with an incomplete IF part but a more complex THEN part. The gradient-descent method can be applied to parameter learning of these systems. Generally, conventional fuzzy systems achieve universal approximation using piecewise-linear functions, while the hierarchical fuzzy system achieves it through piecewise-polynomial functions [114], [68].

4.10 Rule Generation Based on Look-up Table

Designing fuzzy systems from pattern pairs is a nonlinear regression problem. In the simple look-up table (LUT) technique [115], [117], each pattern pair generates one fuzzy rule and then a selection process determines the important rules, which are used to construct the final fuzzy system. In the LUT technique, the input MFs do not change with the sampling data, thus the designed fuzzy system uniformly covers the domain of interest.

In the LUT technique, the input and output spaces are first divided into fuzzy regions, then a fuzzy rule is generated from a given pattern pair, and finally a degree is assigned to each rule to resolve rule conflicts and reduce the number of rules. When the number of examples is large, there is a high probability of conflicting rules, i.e., rules with the same IF parts but different THEN parts. Each rule is assigned a degree of fulfillment. For a group of conflicting rules, only the rule with the maximum degree is retained. When a new pattern pair becomes available, a rule is created for this pattern pair and the fuzzy rule base is updated. The generated rules as well as human expert's knowledge in the form of linguistic rules can be combined so as to produce a

fuzzy rule base. Finally a fuzzy system is built. The LUT technique is implemented in five steps given in [29], [115], [117].

The fuzzy system thus constructed is proved to be a universal approximator by using the Stone-Weierstrass theorem [115]. The approach has the advantage that modification of the rule base is very easy as new examples are available. It is a simple and fast one-pass procedure, since no iterative training is required. Naturally, this algorithm produces an enormous number of rules, when the total input data is considerable. There also arises the problem of contradictory rules, and noisy data in the training examples will affect the consequence of a rule. A similar grid partitioning-based method in which each datum generates one rule has also been derived in [1].

4.11 Other Methods

Many other general methods can be used to automatically extract fuzzy rules from a set of numerical examples and to build a fuzzy system for function approximation; some of these are heuristics-based approaches [42], [92], [28], [105], and hybrid neural-fuzzy approaches such as the ANFIS [48]. In [42], a framework for quickly prototyping an expert system from a set of numerical examples is established. In [92], the fuzzy system can be built in a constructive way. Starting from an initially simple system, the number of MFs in the input domain and the number of rules are adapted in order to reduce the approximation error. A function approximation problem can also be first converted into a pattern classification problem, and then solved by using a fuzzy system [28], [105].

5. FUZZY AND NEURAL: A SYNERGY

While neural networks have strong learning capabilities at the numerical level, it is difficult for the users to understand them at the logic level. Fuzzy logic, on the other hand, has a good capability of interpretability and can also integrate expert's knowledge. The hybridization of both the paradigms yields the capabilities of learning, good interpretation and incorporating prior knowledge. The combination can be in different forms. The simplest form may be the concurrent neuro-fuzzy model, where a fuzzy system and a neural network work separately. The output of one system can be fed as the input of the other system. The cooperative neuro-fuzzy model corresponds to the case that one system is used to adapt the parameters of the other system [38], [94]. The hybrid neural-fuzzy model is the true synergy that captures the merits of both the systems. It takes the form of either a fuzzy neural network or a neuro-fuzzy system. A hybrid neural-fuzzy system does not use multiplication, addition, or the sigmoidal function, but uses fuzzy logic operations such as t-norm and t-conorm.

A fuzzy neural network [84] is a neural network equipped with the capability of handling fuzzy information, where the input signals, activation functions, weights, and/or the operators are based on the fuzzy set theory. Thus, symbolic structure is incorporated. The network can be represented in an equivalent rule-based format, where the premise is the concatenation of fuzzy AND and OR logic, and the consequence is the network output. Two types of fuzzy neurons, namely AND neuron and OR neuron, are defined. The NOT logic is integrated into the weights. Weights always have values in the interval $[0,1]$, and negative weight is achieved by using the NOT operator. The weights of the fuzzy neural network can be interpreted as calibration factors of the conditions and rules. A neuro-fuzzy system is a fuzzy system, whose parameters are learned by a learning algorithm. It has a neural network architecture constructed from fuzzy reasoning, and can always be interpreted as a system of fuzzy rules. Learning is used to adaptively adjust the rules in the rule base, and to produce or optimize the MFs of a fuzzy system. Structured knowledge is codified as fuzzy rules. Expert knowledge can increase learning speed and estimation accuracy. Both fuzzy neural networks and neuro-fuzzy systems can be treated as neural networks, where the units employ the t-norm or t-conorm operator instead of an activation function. The hidden layers represent fuzzy rules. The line between the two hybrid models is blurred, and we call both types of synergisms as neuro-fuzzy systems.

Neuro-fuzzy systems can be obtained by representing some of the parameters of a neural network, such as the inputs, weights, outputs, and shift terms as continuous fuzzy numbers. When only the input is fuzzy, it is a Type I neuro-fuzzy system. When everything except the input is fuzzy, we get a Type II model. A type III model is defined as one where the inputs, weights, and shift terms are all fuzzy. The functions realizing the inference process, such as t-norm and t-conorm, are usually nondifferentiable. To utilize gradient-based algorithms, one has to select differential functions for the inference functions. For nondifferentiable inference functions, training can be performed by using EAs. The shape of the MFs, the number of fuzzy partitions, and rule base can all be evolved by using EAs. The neuro-fuzzy method is superior to the neural network method in terms of the convergence speed and compactness of the structure. Fundamentals in neuro-fuzzy synergism for modeling and control have been reviewed in [51].

5.1 Interpretability

Interpretability is one major reason for using fuzzy systems. Interpretability helps to check the plausibility of a system, leading to easy maintenance of the system. It can also be used to acquire knowledge from a problem characterized by numerical examples. An improvement in interpretability can enhance the performance of generalization when the data set is small. The interpretability of a rule base is usually related to continuity, consistency and completeness [39]. Continuity guarantees that small variations of the input do not induce large variations in the output. Consistency means that if two or more rules are simultaneously fired, their conclusions are coherent. Completeness means that for any possible input vector, at least one rule is fired and there is no inference breaking.

When neuro-fuzzy systems are used to model nonlinear functions described by training sets, the approximation accuracy can be optimized by the learning procedure. However, since learning is accuracy-oriented, it usually causes a reduction in the interpretability of the generated fuzzy system. The loss of interpretability can be due to incompleteness of fuzzy partitions, indistinguishability of fuzzy partitions, inconsistancy of fuzzy rules, too fuzzy or too crisp fuzzy subsets, or incompactness of the fuzzy system [52]. To improve the interpretability of neuro-fuzzy systems, one can add to the cost function, regularization terms that apply constraints on the parameters of fuzzy MFs. For example, the order of the L centers of the fuzzy subset $A^j(x)$, $j = 1, \ldots, L$, should be specified and remain unchanged during learning. Similar MFs should be merged to improve the distinguishability of fuzzy partitions and to reduce the number of fuzzy subsets [96]. One can also reduce the number of free parameters in defining fuzzy subsets. To increase the interpretability of the designed fuzzy system, the same linguistic term should be represented by the same MF. This results in weight sharing [75], [52]. For the TSK model, one practice for good interpretability is to keep the number of fuzzy subsets much smaller than N_r, the number of fuzzy rules, especially when N_r is large.

6. NEURO-FUZZY MODELS

A typical architecture of a neuro-fuzzy system includes an input layer, an output layer, and several hidden layers. The weights are fuzzy sets, and the neurons apply t-norm or t-conorm operations. The hidden layers are usually used as rule layers. The layers before the rule layers perform as premise layers, while those after perform as consequent layers. A well-known neuro-fuzzy model is the ANFIS model [48]. We describe the ANFIS model in this section and also give a brief survey of neuro-fuzzy models.

6.1 ANFIS Model

The ANFIS model [50], [48], [51], as shown in Fig. 7, has a five-layer (n-K-K-K-1) architecture, and is a graphical representation of the TSK model. The functions of the various layers are given below.

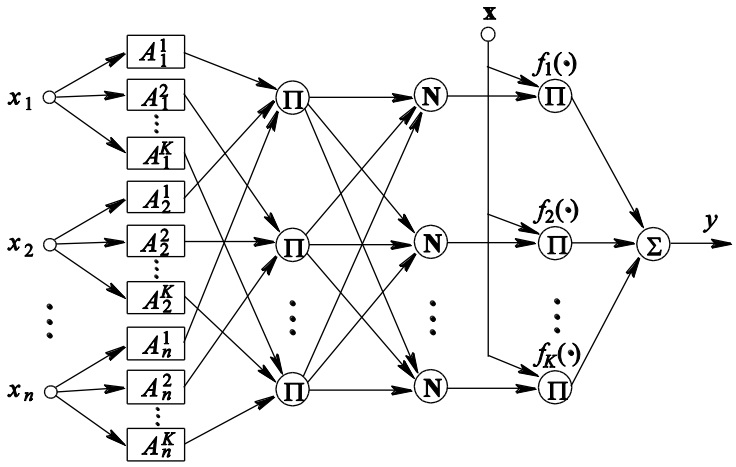


FIGURE 7: ANFIS: graphical representation of the TSK model. The symbol N in the circles denotes the normalization operator, and $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$.

Layer 1 is the input layer with n nodes. The weights between the first two layers, $w_{ij} = \mu_{A_j^i}(x_i)$, $i = 1, \dots, n$, $j = 1, \dots, K$, denotes membership values of the ith input (antecedent) of the jth rule, where A_j^i corresponds to a partition of the space of x_i, and $\mu_{A_j^i}(x_i)$ is typically selected as a generalized bell MF $\mu_{A_j^i}(x_i) = \mu(x_i; c_i^j, a_i^j, b_i^j)$, where c_i^j, a_i^j, and b_i^j are referred to as premise parameters. Layer 2 has K fuzzy neurons with the product t-norm as the aggregation operator. Each node corresponds to a rule, and the output of the jth neuron determines the degree of fulfillment of the jth rule

$$o_j^{(2)} = \prod_{i=1}^{n} \mu_{A_j^i}(x_i) \tag{34}$$

for $j = 1, \dots, K$. Each neuron in layer 3 performs normalization, and the outputs are called *normalized firing strengths*

$$o_j^{(3)} = \frac{o_j^{(2)}}{\sum_{k=1}^{K} o_k^{(2)}} \tag{35}$$

for $j = 1, \dots, K$. The output of each node in layer 4 is defined by

$$o_j^{(4)} = o_j^{(3)} f_j(\mathbf{x}) \tag{36}$$

for $j = 1, \dots, K$. Parameters in $f_j(\mathbf{x})$ are referred to as consequent parameters. The outputs of layer 4 are summed and the output of the network gives the TSK model (30)

$$o^{(5)} = \sum_{j=1}^{K} o_j^{(4)}. \tag{37}$$

In the ANFIS model, functions used at all the nodes are differentiable, thus the BP algorithm can be used to learn the premise parameters by using a sample set of size N, $\{(\mathbf{x}_t, y_t)\}$. The effectiveness of the model is dependent on the MFs used. The TSK fuzzy rules are employed in the ANFIS model

R_i: IF $\mathbf{x}$ is A_i, THEN $y = f_i(\mathbf{x}) = \sum_{j=1}^{n} a_{i,j}\, x_j + a_{i,0}$

for $i = 1, \ldots, K$, where $A_i = \{A_i^1, A_i^2, \ldots, A_i^n\}$ are fuzzy sets and $a_{i,j}, j = 0, 1, \ldots, n$, are consequent parameters. The output of the network at time t is thus given by

$$\hat{y}_t = \frac{\sum_{i=1}^{K} \mu_{A_i}(\mathbf{x}_t) f_i(\mathbf{x}_t)}{\sum_{i=1}^{K} \mu_{A_i}(\mathbf{x}_t)}, \tag{38}$$

where $\mu_{A_i}(\mathbf{x}_t) = \prod_{j=1}^{n} \mu_{A_j^i}(x_{t,j})$. Accordingly, the error measure at time t is defined by $E_t = \frac{1}{2}(\hat{y}_t - y_t)^2$.

After the rule base is specified, the ANFIS adjusts only the MFs of the antecedents and the consequent parameters. The BP algorithm can be used to train both the premise and consequent parameters. A more efficient procedure is to learn the premise parameters by the BP, but to learn the linear consequent parameters by the RLS method [48]. The learning rate η can be adaptively adjusted by some heuristics. It is reported in [48] that this hybrid learning method provides better results than the MLP trained by the BP method and the cascade-correlation network [34]. In [49], the Levenberg-Marquardt (LM) method [29] is used for ANFIS training. Compared to the hybrid method, the LM method achieves a better precision, but the interpretability of the final MFs is quite weak. In [18], the RProp [89] and the RLS methods are used to learn the premise parameters and the consequent parameters, respectively. The ANFIS model has been generalized for classification by employing parameterized t-norms [101], where tree partitioning is used for structure identification and the Kalman filtering method for parameter learning.

The ANFIS is attractive for applications in view of its network structure and the standard learning algorithm. Training of the ANFIS follows the spirit of the minimal disturbance principle and is thus more efficient than the MLP [51]. However, the ANFIS is computationally expensive due to the curse-of-dimensionality problem arising from grid partitioning. Tree or scattering partitioning can resolve the curse of dimensionality, but leads to a reduction in the interpretability of the generated rules. Constraints on MFs and initialization using prior knowledge cannot be provided to the ANFIS model due to the learning procedure. The learning results may be difficult to interpret. Thus, the ANFIS model is suitable for applications, where performance is more important than interpretation. In order to preserve the plausibility of the ANFIS, one can add some regularization terms to the cost function so that some constraints on the interpretability are considered [51].

The ANFIS has been extended to the coactive ANFIS [73] and to the generalized ANFIS [5]. The coactive ANFIS [73] is a generalization of the ANFIS by introducing nonlinearity into the TSK rules. The generalized ANFIS [5] is based on a generalization of the TSK model and a generalized Gaussian RBFN. The generalized fuzzy model is trained by using the generalized RBFN model, based on the functional equivalence between the two models. The sigmoid-ANFIS [125] employs only sigmoidal MFs and adopts the interactive-or operator [7] as its fuzzy connectives. The gradient-descent algorithm can also be directly applied to the TSK model without representing it in a network structure [77]. The unfolding-in-time [119] is a method to transform an RNN into an FNN so that the BP algorithm can be used. The ANFIS-unfolded-in-time [99] is designed for prediction of time series data, and achieves much smaller error in the ANFIS-unfolded-in-time compared to that in the ANFIS.

6.2 Generic Fuzzy Perceptron

The generic fuzzy perceptron (GFP) [75] has a structure similar to that of the three-layer MLP. The network inputs and the weights are modeled as fuzzy sets, and t-norm or t-conorm is used as the activation function at each unit. The hidden layer acts as the rule layer. The output units usually use a defuzzufication function. The GFP can interpret its structure in the form of linguistic rules and the structure of the GFP can be treated as a linguistic rule base, where the weights between the input and hidden (rule) layers are called fuzzy antecedent weights and the weights

between the hidden (rule) and output layers fuzzy consequent weights. The GFP model is based on the Mamdani model.

The NEFCON [76], [75], [78], NEFCLASS [75], and NEFPROX [75] models are three neuro-fuzzy models based on the GFP model, which are used for control, classification and approximation, respectively. Due to the use of nondifferentiable t-norm and t-conorm, the gradient-descent method cannot be applied. A set of linguistic rules are used for describing the performance of the models. This knowledge-based fuzzy error is independent of the range of the output value. Learning algorithms for all these models are derived from the fuzzy error using simple heuristics. Initial fuzzy partitions are needed to be specified for each input variable. Some connections that have identical linguistic values are forced to have the same weights so as to keep the interpretability. Prior knowledge can be integrated in the form of fuzzy rules to initialize the neuro-fuzzy systems, and the remaining rules are obtained by learning.

The NEFCON has a single output node, and is used for control. A reinforcement learning algorithm is used for online learning. The NEFCLASS and the NEFPROX can learn rules by using supervised learning instead of reinforcement learning. Compared to neural networks, the NEFCLASS uses a much simple learning strategy, where no clustering is involved in finding the rules. The NEFCLASS does not use MFs in the rules' consequents. The NETPROX is similar to the NEFCON and the NEFCLASS, but is more general. If there is no prior knowledge, a NEFPROX system can be started with no hidden unit and rules can be incrementally learned. If the learning algorithm creates too many rules, only the best rules are kept by evaluating individual rule errors. Each rule represents a number of samples of the unknown function in the form of fuzzy sample. Parameter learning is used to compensate for the error caused by rule removing.

The NETPROX is more important for function approximation. An empirical performance comparison between the ANFIS and the NETPROX has been made in [75]. The NEFPROX is an order-of-magnitude faster than the ANFIS model of [48], but with a higher approximation error. Interpretation of the learning result is difficult for both the ANFIS and the NEFPROX: the ANFIS represents a TSK system, while the NEFPROX represents a Mamdani system with too many rules. To increase the interpretability of the NEFPROX, pruning strategies can be employed to reduce the number of rules.

6.3 Fuzzy Clustering

Fuzzy clustering is one of the most successful applications of neuro-fuzzy synergism, where fuzzy logic is incorporated into competitive learning-based clustering neural networks such as the Kohonen network and the ART models. In clustering analysis, the discreteness of each cluster endows conventional clustering algorithms with analytical and algorithmic intractabilities. Partitioning the dataset in a fuzzy manner helps to circumvent such difficulties. Each cluster is considered as a fuzzy set, and each feature vector may be assigned to multiple clusters with some degree of certainty measured by the membership function taking values in the interval $[0,1]$. Thus, fuzzy clustering helps to find natural vague boundaries in data. The most well-known fuzzy clustering algorithm is the fuzzy C-means algorithm [8]. Other fuzzy clustering algorithms can be based on the Kohonen network and learning vector quantization, on the ART or the ARTMAP models, or on the Hopfield model. A comprehensive survey on various clustering and fuzzy clustering algorithms is given in [29], [30].

6.4 Other Neuro-Fuzzy Models

Neuro-fuzzy systems can employ the topologies of the layered FNN architecture [40], [53], [23], [26], the RBFN model [2], [120], [79], the self-organizing map (SOM) model [111], and the RNN architecture [65], [66]. Neuro-fuzzy models are mainly used for function approximation. They typically have a layered FNN architecture, are based on TSK-type FISs, and are trained by using the gradient-descent method [82], [46], [40], [64], [54], [67], [108]. Gradient descent in this case is

sometimes termed as the fuzzy BP algorithm. Conjugate gradient (CG) algorithms are also used for training neuro-fuzzy systems [67]. Based on the fuzzification of the linear autoassociative neural networks, the fuzzy PCA [26] can extract a number of relevant features from high-dimensional fuzzy data.

Hybrid neural FIS (HyFIS) [54] is a five-layer neuro-fuzzy model based on the Mamdani FIS. Expert knowledge can be used for the initialization of these MFs. The HyFIS first extracts fuzzy rules from data by using the LUT technique [115]. The gradient-descent method is then applied to tune the MFs of input/output linguistic variables and the network weights by minimizing the error function. The HyFIS model is comparable in performance with the ANFIS [48].

Fuzzy min-max neural networks are a class of neuro-fuzzy models using min-max hyperboxes for clustering, classification, and regression [97], [98], [37], [102], [90]. The max-min fuzzy Hopfield network [66] is a fuzzy RNN for fuzzy associative memory (FAM). The manipulations of the hyperboxes involve mainly comparison, addition and subtraction operations, thus learning is extremely efficient.

Many neuro-fuzzy models employ the architecture of the RBFN [116], [60], [71], [22], [19]. These models use are based on the TSK model, and are a universal approximator. The FBFN can readily adopt various learning algorithms already developed for the RBFN.

Adaptive parsimonious neuro-fuzzy systems can be achieved by using constructive approach and a simultaneous adaptation of space partitioning and fuzzy rule parameters [22], [120]. The dynamic fuzzy neural network (DFNN) [120], [33] is an online implementation of the TSK system based on an extended RBFN and its learning algorithm. Similar to the ANFIS architecture, the self-organizing fuzzy neural network (SOFNN) [62] has a five-layer fuzzy neural network architecture. It is an online implementation of a TSK-type model. The SOFNN is based on neurons with an ellipsoidal basis function, and the neurons are added or pruned dynamically in the learning process. Similar MFs can be combined into one new MF. The SOFNN algorithm is superior to the DFNN in time complexity [120].

7. FUZZY NEURAL CIRCUITS

Fuzzy systems can be easily implemented in the digital form, which can be either general-purpose microcontrollers running fuzzy inference and defuzzification programs, or dedicated fuzzy coprocessors, or RISC processors with specialized fuzzy support, or fuzzy ASICs. The pros and cons of various digital fuzzy hardware implementation strategies are reviewed in [25].

A common approach to general-purpose fuzzy hardware is to use a software design tool such as the Motorola-Aptronix fuzzy inference development language and Togai InfraLogic's MicroFPL system to generate the program code for a target microcontroller [44]. This approach leads to rapid design and testing, but has a low performance. On the other hand, dedicated fuzzy processors and ASICs have physical and performance characteristics that are closely matched to an application, and its performance would be optimized to suit a given problem at the price of high design and test costs. Fuzzy coprocessors work in conjunction with a host processor. They are general-purpose hardware, and thus have a lower performance compared to a custom fuzzy hardware. A number of commercially available fuzzy coprocessors are listed in [95]. Some issues arising from the implementation of such coprocessors are discussed in [83]. RISC processors with specialized fuzzy support are also available [25], [95]. A fuzzy-specific extension to the instruction set is defined and implemented using hardware/software codesign techniques. In [44], the tool TROUT was created to automate fuzzy neural ASIC design. The TROUT produces a specification for small, application-specific circuits called smart parts. Each smart part is customized to a single function, and can be packaged in a variety of ways. The model library of the TROUT includes fuzzy or neural models for implementation as circuits. To synthesize a circuit,

the TROUT takes as its input an application data set, optionally augmented with user-supplied hints. It delivers, as output, technology-independent VHDL code for a circuit of the fuzzy or neural model.

There are also many analog [61], [24], [58], and mixed-signal [6], [10] fuzzy circuits. Analog circuits usually operate in the current mode and are fabricated using the CMOS technology, and this leads to the advantages of high speed, small-circuit area, high performance, and low power dissipation. A design methodology for fuzzy ASICs and general-purpose fuzzy processors is given in [58], based on the LR (left-right) fuzzy implication cells and the LR fuzzy arithmetic cells. In [6], [10], the fabrication of mixed-signal CMOS chips for fuzzy controllers is considered; in these circuits, the computing power is provided by the analog part while the digital part is used for programmability.

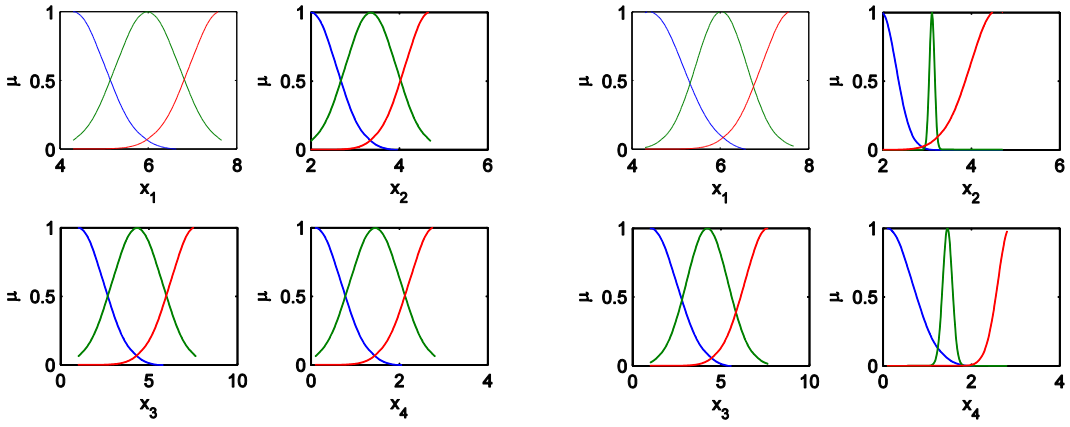
An overview of the existing hardware implementations of neural and fuzzy systems is made in [88], where limitations, advantages, and bottlenecks of analog, digital, pulse stream (spiking), and other techniques are discussed. Hardware/software codesign allows a fast design of complex systems with the highest performance-cost ratio by exploiting the best from both the hardware and software techniques. A survey of digital fuzzy logic controllers is given in [83].

8. COMPUTER SIMULATION: IRIS CLASSIFICATION

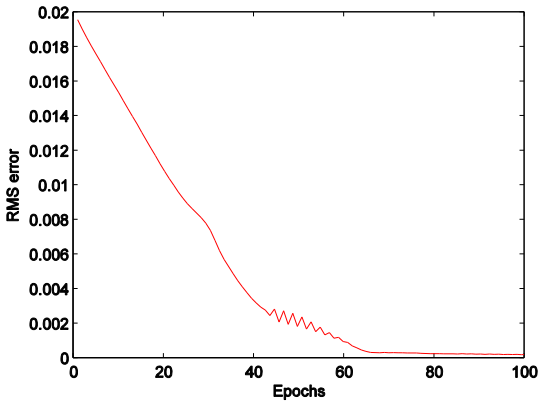
We now use the ANFIS model to solve the Iris classification problem. In the Iris data set, 150 patterns are classified into 3 classes. Each pattern has four numeric properties. The ANFIS model is available in the MATLAB Fuzzy toolbox. An initial TSK FIS is first generated by using grid partitioning. Since the ranges for x_1, x_2, and x_3 are very small, they each are partitioned into 2 subsets. The Gaussian MF is selected.

We use the ANFIS model to solve the IRIS classification problem. For the 120 patterns, the ranges of the input and output variables are $x_1 \in [4.3, 7.9]$, $x_2 \in [2.0, 4.4]$, $x_3 \in [1.0, 6.9]$, $x_4 \in [0.1, 2.5]$, $y \in [1, 3]$.

An initial TSK FIS is first generated by using grid partitioning. The variables each are partitioned into 3 subsets. The Gaussian MF is selected. The maximum number of epochs is 100. The fuzzy partitioning for the input space as well as the training error is illustrated in Fig. 8. The classification error rate is 0. The ANFIS model generates 193 nodes, 405 linear parameters, 24 nonlinear parameters, and 81 fuzzy rules. The training time is 53.70 s.



(a) (b)

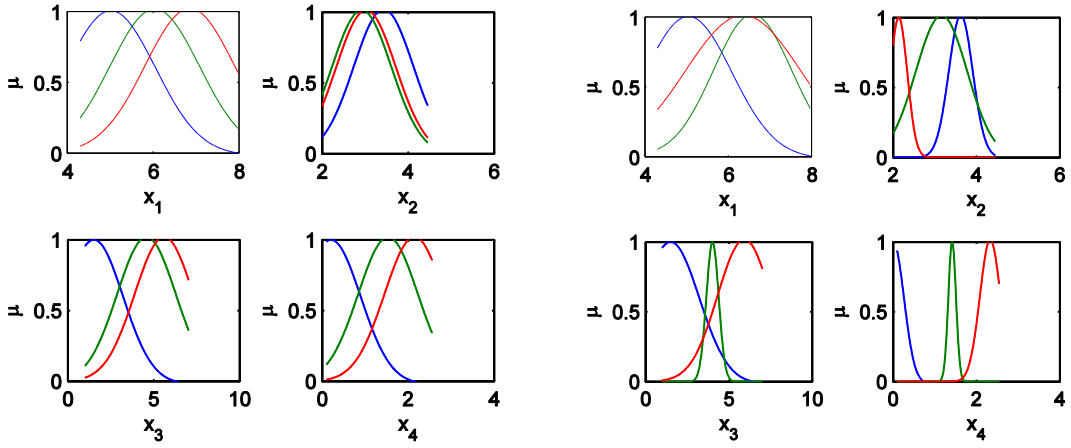


(c)

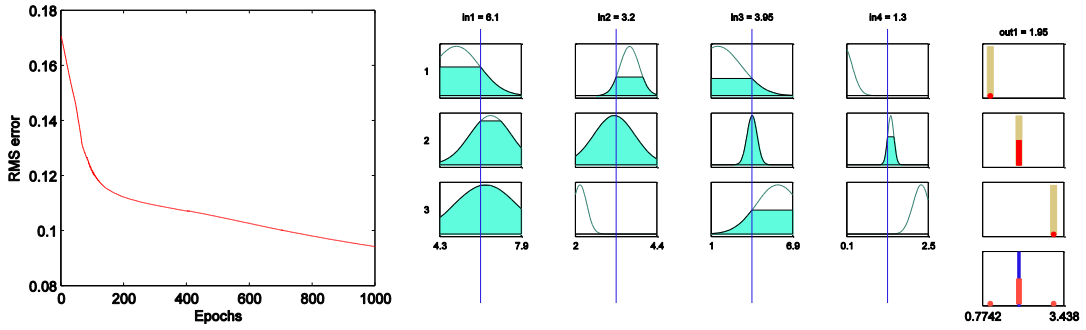
FIGURE 8: IRIS classification: grid partitioning of the input space. (a) The initialized MFs. (b) The learned MFs. (c) The training RMS error.

We further solve the IRIS problem using the ANFIS with scatter partitioning. Clustering the input space is a desired method for generating fuzzy rules. This can significantly reduce the total number of fuzzy rules, hence offer a better generalization capability. Subtractive clustering is used for rule extraction so as to find an initial FIS for ANFIS training. Radius r specifies the range of influence of the cluster center for each input and output dimension. The training error can be controlled by adjusting r, $r \in [0,1]$. Specifying a smaller cluster radius usually yields more, smaller clusters in the data, and hence more rules. The training runs for 200 epochs.

Since the range of the input space is very small when compared with that of the output space, we select $r = 0.8$ for all the input dimensions and the output space. The training time is 2.69 s. After training the RMS error is 0.1123. The ANFIS model has 37 nodes, 15 linear parameters, 24 nonlinear parameters, and 3 fuzzy rules. The classification error is 1.33%. The scatter partitioning is shown in Fig. 9a, b, and the training and testing errors are illustrated in Fig. 9c. The generated fuzzy rules are shown in Fig. 9d.



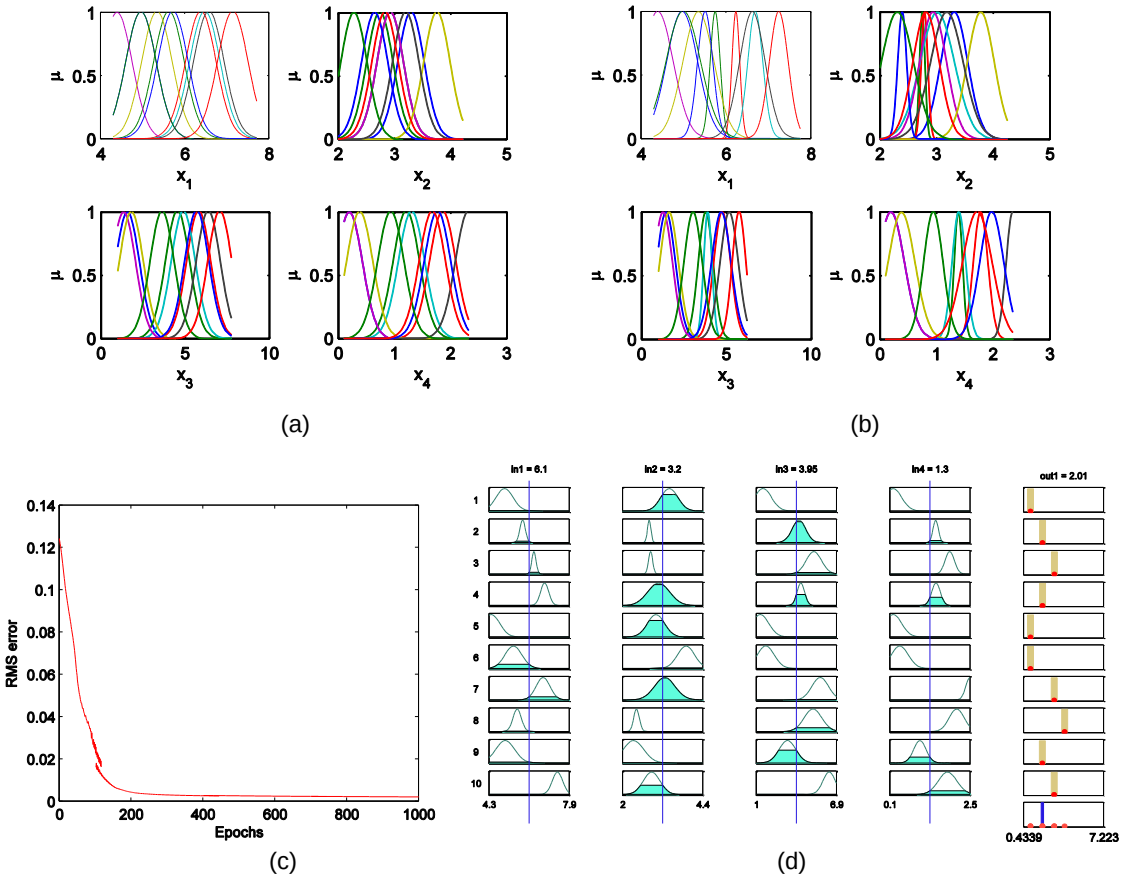
(a) (b)



(c) (d)

FIGURE 9: IRIS classification: scatter partitioning of the input space. (a) The initialized MFs. (b) The learned MFs. (c) The training RMS error. (d) the generated fuzzy rules. Note that some MFs coincide in the figure. $\mathbf{r} = [0.8, 0.8, 0.8, 0.8, 0.8]$.

In order to further increase the training accuracy, we can select $r = 0.3$ for all the input dimensions and the output space to get a finer clustering. Then we can get more rules. The ANFIS model has 107 nodes, 50 linear parameters, 80 nonlinear parameters, and 19 fuzzy rules. The training time is 16.2624 s for 1000 epochs. The result is shown in Fig. 10.



(a) (b)

(c) (d)

FIGURE 10: IRIS classification: scatter partitioning of the input space. (a) The initialized MFs. (b) The learned MFs. (c) The training RMS error. (d) the generated fuzzy rules. Note that some MFs coincide in the figure. $\mathbf{r} = [0.9, 0.9, 0.9, 0.9, 0.1]$.

For the 10 rules generated, each rule has its own MF for each input variable. For example, the ith rule is given by

$$\mathrm{R}_i\text{: IF } x_1 \text{ is } \mu_{i,1} \text{ AND } x_2 \text{ is } \mu_{i,2} \text{ AND } x_3 \text{ is } \mu_{i,3} \text{ AND } x_4 \text{ is } \mu_{i,4} \text{ THEN } y \text{ is } \mu_{i,y}$$

where $\mu_{i,k}$, $k = 1, \ldots, 4$, and $\mu_{i,y}$ are MFs. The fuzzy rules for the DoA estimation using the ANFIS with scattering partitioning and the fuzzy-inference process from inputs to outputs. Each row of plots corresponds to one rule, and each column corresponds to either an input variable x_i or the output variable y.

9. SUMMARY

In this paper, we give a systematic introduction to concepts in fuzzy sets and fuzzy logic as well as neuro-fuzzy systems. Fuzzy logic provides an effective tools for modelling uncertainty in human reasoning. A fuzzy inference system represents knowledge in IF-THEN rules, and implement fuzzy reasoning. Like neural network models, some fuzzy inference systems have the universal approximation capability. Fuzzy logic is an alternative to neural networks for the purpose of classification and function approximation and for most applications where neural networks are applicable. Neuro-fuzzy systems combine the advantages of both computational paradigms, and are gaining more popularity.

10. REFERENCES

1. S. Abe, M.S. Lan. “Fuzzy rules extraction directly from numerical data for function approximation”. IEEE Trans. Syst. Man Cybern., 25(1), pp. 119–129, 1995.

2. D.F. Akhmetov, Y. Dote, S.J. Ovaska. “Fuzzy neural network with general parameter adaptation for modeling of nonlinear time-series”. IEEE Trans. Neural Netw., 12(1), pp. 148–152, 2001.

3. J.S. Albus. “A new approach to manipulator control: Cerebellar model articulation control (CMAC)”. Trans. ASME J. Dyna. Syst. Meas. Contr., 97, pp. 220–227, 1975.

4. P.P. Angelov, D.P. Filev. “An approach to online identification of Takagi-Sugeno fuzzy models”. IEEE Trans. Syst. Man Cybern. B, 34(1), 484–498, 2004.

5. M.F. Azeem, M. Hanmandlu, N. Ahmad. “Generalization of adaptive neuro-fuzzy inference systems”. IEEE Trans. Neural Netw., 11(6), pp. 1332–1346, 2000.

6. I. Baturone, S. Sanchez-Solano, A. Barriga, J.L. Huertas. “Implementation of CMOS fuzzy controllers as mixed-signal integrated circuits”. IEEE Trans. Fuzzy Syst., 5(1), pp. 1–19, 1997.

7. J.M. Benitez, J.L. Castro, I. Requena. “Are artificial neural networks black boxes?”. IEEE Trans. Neural Netw., 8(5), pp. 1156–1164, 1997.

8. J. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms, New York: Plenum Press, 1981.

9. A. Blanco, M. Delgado, M.C. Pegalajar. “Extracting rules from a (fuzzy/crisp) recurrent neural network using a self-organizing map”. Int. J. Intell. Syst., 15(7), pp. 595–621, 2000.

10. S. Bouras, M. Kotronakis, K. Suyama, Y. Tsividis. “Mixed analog-digital fuzzy logic controller with continuous-amplitude fuzzy inferences and defuzzification”. IEEE Trans. Fuzzy Syst., 6(2), pp. 205–215, 1998.

11. J.J. Buckley. “Fuzzy complex numbers”. Fuzzy Sets Syst., 33, pp. 333–345, 1989.

12. J.J. Buckley. “Sugeno type controllers are universal controllers”. Fuzzy Sets Syst., 53, pp. 299–304, 1993.

13. J.J. Buckley, Y. Hayashi, E. Czogala. “On the equivalence of neural nets and fuzzy expert systems”. Fuzzy Sets Syst., 53, pp. 129–134, 1993.

14. J.J. Buckley, E. Eslami. An Introduction to Fuzzy Logic and Fuzzy Sets, Heidelberg: Physica-Verlag, 2002.

15. J.L. Castro. “Fuzzy logic controllers are universal approximators”. IEEE Trans Syst Man Cybern., 25(4), pp. 629–635, 1995.

16. J.L. Castro, C.J. Mantas, J. Benitez. “Interpretation of artificial neural networks by means of fuzzy rules”. IEEE Trans. Neural Netw., 13(1), pp. 101–116, 2002.

17. A. Cechin, U. Epperlein, B. Koppenhoefer, W. Rosenstiel. “The extraction of Sugeno fuzzy rules from neural networks”, in Proc. Euro. Symp. Artif. Neural Netw., Bruges, Belgium, 1996, pp. 49–54.

18. M.S. Chen, R.J. Liou. “An efficient learning method of fuzzy inference system”, in Proc. IEEE Int. Fuzzy Syst. Conf., Seoul, Korea, 1999, pp. 634–638.

19. C.B. Cheng, E.S. Lee. “Fuzzy regression with radial basis function network”. Fuzzy Sets Syst., 119, pp. 291–301, 2001.

20. S. Chiu. “Fuzzy model identification based on cluster estimation”. J. Intell. & Fuzzy Syst., 2(3), pp. 267–278, 1994.

21. S.L. Chiu. “A cluster estimation method with extension to fuzzy model identification”, in Proc. IEEE Int. Conf. Fuzzy Syst., Orlando, FL, 2, 1994, pp. 1240–1245.

22. K.B. Cho, B.H. Wang. “Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction”. Fuzzy Sets Syst., 83, pp. 325–339, 1996.

23. M.Y. Chow, S. Altrug, H.J. Trussell. “Heuristic constraints enforcement for training of and knowledge extraction from a fuzzy/neural architecture—Part I: Foundations”. IEEE Trans. Fuzzy Syst., 7(2), pp. 143–150, 1999.

24. U. Cilingiroglu, B. Pamir, Z.S. Gunay, F. Dulger. “Sampled-analog implementation of application-specific fuzzy controllers”. IEEE Trans. Fuzzy Syst., 5(3), pp. 431–442, 1997.

25. A. Costa, A. De Gloria, P. Farabosch, A. Pagni, G. Rizzotto. “Hardware solutions of fuzzy control”. Proc. IEEE, 83(3), pp. 422–434, 1995.

26. T. Denoeux, M.H. Masson. “Principal component analysis of fuzzy data using autoassociative neural networks”. IEEE Trans. Fuzzy Syst., 12(3), pp. 336–349, 2004.

27. S. Dick. “Toward complex fuzzy logic”. IEEE Trans. Fuzzy Syst., 13(3), pp. 405–414, 2005.

28. J.A. Dickerson, B. Kosko. “Fuzzy function learning with covariance ellipsoids”, in Proc. IEEE Int. Conf. Neural Netw., San Francisco, 3, 1993, pp. 1162–1167.

29. K.-L. Du, M.N.S. Swamy. Neural Networks in a Softcomputing Framework, London: Springer, 2006.

30. K.-L. Du. “Clustering: a neural network approach”. Neural Netw., 23(1), pp. 89–107, 2010.

31. C. Dualibe, M. Verleysen, P.G.A. Jespers. Design of Analog Fuzzy Logic Controllers in CMOS Technology, Netherlands: Kluwer, 2003.

32. W. Duch. “Uncertainty of data, fuzzy membership functions, and multilayer perceptrons”. IEEE Trans. Neural Netw., 16(1), pp. 10–23, 2005.

33. M.J. Er, S. Wu. “A fast learning algorithm for parsimonious fuzzy neural systems”. Fuzzy Sets Syst., 126, pp. 337–351, 2002.

34. S.E. Fahlman, C. Lebiere. “The cascade-correlation learning architecture”, in Advances in Neural Information Processing Systems 2, D.S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524–532.

35. G. Ferrari-Trecate, R. Rovatti. “Fuzzy systems with overlapping Gaussian concepts: Approximation properties in Sobolev norms”. Fuzzy Sets Syst., 130, pp. 137–145, 2002.

36. M. Figueiredo, F. Gomides, A. Rocha, R. Yager. “Comparison of Yager’s level set method for fuzzy logic control with Mamdani and Larsen methods”. IEEE Trans. Fuzzy Syst., 2, pp. 156–159, 1993.

37. B. Gabrays, A. Bargiela. “General fuzzy min-max neural networks for clustering and classification”. IEEE Trans. Neural Netw., 11(3), pp. 769–783, 2000.

38. S.I. Gallant. “Connectionist expert systems”. Commun. of ACM, 31(2), pp. 152–169, 1988.

39. S. Guillaume. “Designing fuzzy inference systems from data: An interpretability-oriented review”. IEEE Trans. Fuzzy Syst., 9(3), pp. 426–443, 2001.

40. Y. Hayashi, J.J. Buckley, E. Czogala. “Fuzzy neural network with fuzzy signals and weights”. Int. J. Intell. Syst., 8(4), pp. 527–537, 1993.

41. P. Holmblad, J. Ostergaard. “Control of a cement kiln by fuzzy logic”, in Fuzzy Information and Decision Processes, M.M. Gupta, E. Sanchez, Eds. Amsterdam: North-Holland, 1982, pp. 389–399.

42. T. Hong, C. Lee. “Induction of fuzzy rules and membership functions from training examples”. Fuzzy Sets Syst., 84, pp. 33–37, 1996.

43. K.J. Hunt, R. Haas, R. Murray-Smith. “Extending the functional equivalence of radial basis function networks and fuzzy inference systems”. IEEE Trans. Neural Netw., 7(3), pp. 776–781, 1996.

44. J.F. Hurdle. “The synthesis of compact fuzzy neural circuits”. IEEE Trans. Fuzzy Syst., 5(1), pp. 44–55, 1997.

45. M. Ishikawa. “Rule extraction by successive regularization”. Neural Netw., 13(10), pp. 1171–1183, 2000.

46. H. Ishibuchi, R. Fujioka, H. Tanaka. “Neural networks that learn from fuzzy IF-THEN rules”. IEEE Trans. Fuzzy Syst., 1, pp. 85–97, 1993.

47. H. Jacobsson. “Rule extraction from recurrent neural networks: A taxonomy and review”. Neural Comput., 17(6), pp. 1223–1263, 2005.

48. J.S.R. Jang. “ANFIS: Adaptive-network-based fuzzy inference systems”. IEEE Trans, Syst, Man Cybern., 23(3), pp. 665–685, 1993.

49. J.S.R. Jang, E. Mizutani. “Levenberg-Marquardt method for ANFIS learning”, in Proc. Biennial Conf. North Amer. Fuzzy Inf. Process. Soc. (NAFIPS), Berkeley, CA, 1996, pp. 87–91.

50. J.S.R. Jang, C.I. Sun. “Functional equivalence between radial basis function Networks and fuzzy inference systems”. IEEE Trans. Neural Netw., 4(1), pp. 156–159, 1993.

51. J.S.R. Jang, C.I. Sun. “Neuro-fuzzy modeling and control”. Proc. IEEE, 83(3), pp. 378–406, 1995.

52. Y. Jin. “Advanced fuzzy systems design and applications”. Heidelberg: Physica-Verlag, 2003.

53. C.F. Juang, C.T. Lin. “An on-line self-constructing neural fuzzy inference network and its application”. IEEE Trans. Fuzzy Syst., 6(1), pp. 12–32, 1998.

54. J. Kim, N. Kasabov. “HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems”. Neural Netw., 12, pp. 1301–1319, 1999.

55. B. Kosko. “Fuzzy system as universal approximators”, in Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, CA, 1992, pp. 1153–1162.

56. B. Kosko. Fuzzy engineering. Prentice Hall, Englewood Cliffs, 1997.

57. V. Kreinovich, H.T. Nguyen, Y. Yam. “Fuzzy systems are universal approximators for a smooth function and its derivatives”. Int. J. Intell. Syst., 15, pp. 565–574, 2000.

58. Y.H. Kuo, C.L. Chen. “Generic LR fuzzy cells for fuzzy hardware synthesis”. IEEE Trans. Fuzzy syst., 6(2), pp. 266–285, 1998.

59. R. Langari, L. Wang, J. Yen. “Radial basis function networks, regression weights, and the expectation-maximization algorithm”. IEEE Trans. Syst. Man Cybern. A, 27(5), 613–623, 1997.

60. C.W. Lee, Y.C. Shin. “Construction of fuzzy systems using least-squares method and genetic algorithm”. Fuzzy Sets Syst., 137, pp. 297–323, 2003.

61. L. Lemaitre, M. Patyra, D. Mlynek. “Analysis and design of CMOS fuzzy logic controller in current mode”. IEEE J. Solid-State Circ., 29(3), pp. 317–322, 1994.

62. G. Leng, G. Prasad, T.M. McGinnity. “An on-line algorithm for creating self-organizing fuzzy neural networks”. Neural Netw., 17, pp. 1477–1493, 2004.

63. H.X. Li, C.L.P. Chen. “The equivalence between fuzzy logic systems and feedforward neural networks”. IEEE Trans. Neural Netw., 11(2), pp. 356–365, 2000.

64. C.T. Lin, Y.C. Lu. “A neural fuzzy system with fuzzy supervised learning”. IEEE Trans. Syst. Man Cybern. B, 26(5), pp. 744–763, 1996.

65. F.J. Lin, R.J. Wai. “Hybrid control using recurrent fuzzy neural network for linear-induction motor servo drive”. IEEE Trans. Fuzzy Syst., 9(1), pp. 102–115, 2001.

66. P. Liu. “Max-min fuzzy Hopfield neural networks and an efficient learning algorithm”. Fuzzy Sets Syst., 112, pp. 41–49, 2000.

67. P. Liu, H. Li. “Efficient learning algorithms for three-layer regular feedforward fuzzy neural networks”. IEEE Trans. Neural Netw., 15(3), 545–558, 2004.

68. P. Liu, H. Li. “Hierarchical TS fuzzy system and its universal approximation”. Inf. Sci., 169, pp. 279–303, 2005.

69. E.H. Mamdani. “Application of fuzzy algorithms for control of a simple dynamic plant”. Proc. IEEE, 12(1), pp. 1585–1588, 1974.

70. K.J. McGarry, J. MacIntyre. “Knowledge extraction and insertion from radial basis function networks”, in IEE Colloq. Applied Stat. Pattern Recogn, Birmingham, UK, 1999, pp. 15/1–15/6

71. S. Mitra, J. Basak. “FRBF: A fuzzy radial basis function network”. Neural Comput. & Appl., 10, pp. 244–252, 2001.

72. S. Mitra, Y. Hayashi. “Neuro-fuzzy rule generation: Survey in soft computing framework”. IEEE Trans. Neural Netw., 11(3), pp. 748–768, 2000.

73. E. Mizutani, J.S. Jang. “Coactive neural fuzzy modeling”, in Proc. IEEE Int. Conf. Neural Netw., Perth, Australia, 1995, vol. 2, pp. 760–765.

74. D. Moses, O. Degani, H.N. Teodorescu, M. Friedman, A. Kandel. “Linguistic coordinate transformations for complex fuzzy sets”, in Proc. IEEE Int. Conf. Fuzzy Syst., Seoul, Korea, 1999, 3, pp. 1340–1345.

75. D. Nauck, F. Klawonn, R. Kruse. Foundations of Neuro-fuzzy Systems. New York: Wiley, 1997.

76. D. Nauck, R. Kruse. “A neural fuzzy controller learning by fuzzy error propagation”, in Proc. Worksh. North Amer. Fuzzy Inf. Process. Soc. (NAFIPS92), Puerto Vallarta, Mexico, 1992, pp. 388–397.

77. H. Nomura, I. Hayashi, N. Wakami. “A learning method of fuzzy inference rules by descent method”, in Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, CA, 1992, pp. 203–210.

78. A. Nurnberger, D. Nauck, R. Kruse. “Neuro-fuzzy control based on the NEFCON-model: Recent developments”. Soft Comput., 2, pp. 168–182, 1999.

79. S.K. Oh, W. Pedrycz, H.S. Park. “Multi-layer hybrid fuzzy polynomial neural networks: A design in the framework of computational intelligence”. Neurocomput., 64, pp. 397–431, 2005.

80. C.W. Omlin, C.L. Giles. “Extraction of rules from discrete-time recurrent neural networks”. Neural Netw., 9, pp. 41–52, 1996.

81. C.W. Omlin, K.K. Thornber, C.L. Giles. “Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks”. IEEE Trans. Fuzzy Syst., 6, pp. 76–89, 1998.

82. S.K. Pal, S. Mitra. “Multilayer perceptron, fuzzy sets, and classification”. IEEE Trans. Neural Netw., 3(5), pp. 683–697, 1992.

83. M.J. Patyra, J.L. Grantner, K. Koster. “Digital fuzzy logic controller: Design and implementation”. IEEE Trans. Fuzzy Syst., 4(4), pp. 439–459, 1996.

84. W. Pedrycz, A.F. Rocha. “Fuzzy-set based models of neurons and knowledge-based networks”. IEEE Trans. Fuzzy Syst., 1(4), pp. 254–266, 1993.

85. G.V.S. Raju, J. Zhou, R.A. Kisner. “Hierarchical fuzzy control”. Int. J. Contr., 54(5), pp. 1201–1216, 1991.

86. D. Ramot, M. Friedman, G. Langholz, A. Kandel. “Complex fuzzy logic”. IEEE Trans. Fuzzy Syst., 11(4), pp. 450–461, 2003.

87. D. Ramot, R. Milo, M. Friedman, A. Kandel. “Complex fuzzy sets”. IEEE Trans. Fuzzy Syst., 10(2), pp. 171–186, 2002.

88. L.M. Reyneri. “Implementation issues of neuro-fuzzy hardware: Going toward HW/SW codesign”. IEEE Trans. Neural Netw., 14(1), pp. 176–194, 2003.

89. M. Riedmiller, H. Braun. “A direct adaptive method for faster backpropagation learning: the RPROP algorithm”, in Proc. IEEE Int. Conf. Neural Netw., San Francisco, CA, 1993, pp. 586–591.

90. A. Rizzi, M. Panella, F.M.F. Mascioli. “Adaptive resolution min-max classifiers”. IEEE Trans. Neural Netw., 13(2), pp. 402–414, 2002.

91. I. Rojas, H. Pomares, J.L. Bernier, J. Ortega et al. “Time series analysis using normalized PG-RBF network with regression weights”. Neurocomput., 42, pp. 267–285, 2002.

92. I. Rojas, H. Pomares, J. Ortega, A. Prieto. “Self-organized fuzzy system generation from training examples”. IEEE Trans. Fuzzy Syst., 8(1), pp. 23–36, 2000.

93. M. Russo. “Fugenesys—A fuzzy genetic neural system for fuzzy modeling”. IEEE Trans. Fuzzy Syst., 6(3), pp. 373–388, 1998.

94. K. Saito, R. Nakano. “Rule extraction from facts and neural networks”, in Proc. Int. Neural Netw. Conf., Paris, France, pp. 379–382. Kluwer, Dordrecht, the Netherland, 1990.

95. V. Salapura. “A fuzzy RISC processor”. IEEE Trans. Fuzzy Syst., 8(6), pp. 781–790, 2000.

96. M. Setnes, R. Babuska, U. Kaymak, H.R. van Nauta Remke. “Similarity measures in fuzzy rule base simplification”. IEEE Trans. Syst. Man Cybern. B, 28(3), pp. 376–386, 1998.

97. P.K. Simpson. “Fuzzy min-max neural networks–Part I. classification”. IEEE Trans. Neural Netw., 3, pp. 776–786, 1992.

98. P.K. Simpson. “Fuzzy min-max neural networks–Part II: clustering”. IEEE Trans. Fuzzy Syst., 1(1), pp. 32–45, 1993.

99. N.A. Sisman-Yilmaz, F.N. Alpaslan, L. Jain. “ANFIS-unfolded-in-time for multivariate time series forecasting”. Neurocomput., 61, pp. 139–168, 2004.

100. E. Soria-Olivas, J.D. Martin-Guerrero, G. Camps-Valls, A.J. Serrano-Lopez, J. Calpe-Maravilla, L. Gomez-Chova. “A low-complexity fuzzy activation function for artificial neural networks”. IEEE Trans. Neural Netw., 14(6), pp. 1576–1579, 2003.

101. C.T. Sun. “Rule-base structure identification in an adaptive-network-based inference system”. IEEE Trans. Fuzzy Syst., 2(1), pp. 64–79, 1994.

102. R. Tagliaferri, A. Eleuteri, M. Meneganti, F. Barone. “Fuzzy min-max neural networks: From classification to regression”. Soft Comput., 5, pp. 69–76, 2001.

103. T. Takagi, M. Sugeno. “Fuzzy identification of systems and its applications to modelling and control”. IEEE Trans. Syst. Man Cybern., 15(1), pp. 116–132, 1985.

104. K. Tanaka, M. Sugeno. “Stability analysis and design of fuzzy control systems”. Fuzzy Sets Syst., 45, pp. 135–150, 1992.

105. R. Thawonmas, S. Abe. “Function approximation based on fuzzy rules extracted from partitioned numerical data”. IEEE Trans. Syst. Man Cybern. B, 29(4), pp. 525–534, 1999.

106. A. Tickle, R. Andrews, M. Golea, J. Diederich. “The truth will come to light: Direction and challenges in extracting the knowledge embedded within trained artificial neural networks”. IEEE Trans. Neural Netw., 9(6), pp. 1057–1068, 1998.

107. V. Tresp, J. Hollatz, S. Ahmad. “Representing probabilistic rules with networks of Gaussian basis functions”. Mach. Learn., 27, pp. 173–200, 1997.

108. G. Tsekouras, H. Sarimveis, E. Kavakli, G. Bafas. “A hierarchical fuzzy-clustering approach to fuzzy modeling”. Fuzzy Sets Syst., 150(2), pp. 245–266, 2004.

109. K. Uehara, M. Fujise. “Fuzzy inference based on families of α-level sets”. IEEE Trans. Fuzzy Syst., 1(2), pp. 111–124, 1993.

110. A. Ultsch, R. Mantyk, G. Halmans. “Connectionist knowledge acquisition tool: CONKAT”, in Artificial Intelligence Frontiers in Statistics: AI and Statistics III, : D.J. Hand, Ed. London: Chapman & Hall, 1993, pp. 256–263.

111. P. Vuorimaa. “Fuzzy self-organizing map”. Fuzzy Sets Syst., 66(2), pp. 223–231, 1994.

112. D. Wang, N.S. Chaudhari. “Binary neural network training algorithms based on linear sequential learning”. Int J. Neural Syst., 13(5), pp. 333–351, 2003.

113. L.X. Wang. “Fuzzy systems are universal approximators”, in Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, CA, 1992, pp. 1163–1170.

114. L.X. Wang. “Analysis and design of hierarchical fuzzy systems”. IEEE Trans. Fuzzy Syst., 7(5), pp. 617–624, 1999.

115. L.X. Wang, J.M. Mendel. “Generating fuzzy rules by learning from examples”. IEEE Trans. Syst. Man Cybern., 22(6), pp. 1414–1427, 1992.

116. L.X. Wang, J.M. Mendel. “Fuzzy basis functions, universal approximation, and orthogonal least-squares learning”. IEEE Trans. Neural Netw., 3(5), pp. 807–814, 1992.

117. L.X. Wang, C. Wei. “Approximation accuracy of some neuro-fuzzy approaches”. IEEE Trans. Fuzzy Syst., 8(4), pp. 470–478, 2000.

118. S.Wang, H. Lu. “Fuzzy system and CMAC network with B-spline membership/basis functions are smooth approximators”. Soft Comput., 7, pp. 566–573, 2003.

119. P.J. Werbos. “Backpropagation through time: what it does and how to do it”. Proc. IEEE, 78(10), pp. 1550–1560, 1990.

120. S. Wu, M.J. Er. “Dynamic fuzzy neural networks—A novel approach to function approximation”. IEEE Trans. Syst. Man Cybern. B, 30(2), pp. 358–364, 2000.

121. R. Yager, D. Filev. “Generation of fuzzy rules by mountain clustering”. J. Intell. Fuzzy Syst., 2(3), pp. 209–219, 1994.

122. J. Yen. “Fuzzy logic—A modern perspective”. IEEE Trans. Knowl. Data Eng., 11(1), pp. 153–165, 1999.

123. L.A. Zadeh. “Fuzzy sets”. Inf. & Contr., 8, pp. 338–353, 1965.

124. L.A. Zadeh. “The concept of a linguistic variable and its application to approximate reasoning–I, II, III”. Inf. Sci., 8, pp. 199–249, pp. 301–357, 1975; 9, pp. 43–80, 1975.

125. D. Zhang, X.L. Bai, K.Y. Cai. “Extended neuro-fuzzy models of multilayer perceptrons”. Fuzzy Sets Syst., 142, pp. 221–242, 2004.