ELSEVIER

Fuzzy Sets and Systems 82 (1996) 161–166

FUZZY sets and systems

Rule combination in a fuzzy neural network

Young-Jeng Chen *, Ching-Cheng Teng

National Chiao-Tung University, Institute of Control Engineering, Hsinchu, Taiwan

Abstract

In this paper, we present a fuzzy neural network (FNN) to realize the rule reasoning of fuzzy inference systems. The proposed fuzzy neural network can acquire the fuzzy rules by employing the learning capability of neural networks. Moreover, for simplifying the structure of the proposed FNN, the redundant rules and linguistic terms are removed from the FNN by means of a rule combination procedure. A sufficient condition related to rule combination is also given.

Keywords: Fuzzy neural network; Fuzzy inference system; Rule combination

1. Introduction

The main goal of a fuzzy inference system is to model human decision making within the conceptual framework of fuzzy logic and approximate reasoning. Basically, fuzzy inference system uses fuzzy if–then rules to infer stipulated relationship for input–output pairs of a system. From a system theoretic point of view, this task merely try to get satisfied nonlinear static mapping for a system. Many kinds of fuzzy modeling techniques have been developed, e.g., [1, 2, 4, 5, 7]. As we saw in many literature [1, 5], the initial state of fuzzy rules is obtained after carefully partition the input space associated with each input variable. The fuzzy rules in this case are *complete*, since every partition region of the input space performs a fuzzy rule. But, by making this kind of partition, the number of rules increases exponentially with the number of inputs. Moreover, there may exist some redundant rules among all of the complete fuzzy rules. In other words, some rules should be expressed as a single rule. A procedure for finding such a single rule is usually termed *rule combination*. Lin and Lee [5] proposed a criterion of rule combination for reducing the number of rules in their connectionist fuzzy logic control/decision system. To express the rule combination in a mathematical way according to their structure, however, has not yet been understood clearly.

In this paper, the FNN, a special type of fuzzy neural networks is presented. Also, the structure of the FNN will be shown as having the capability to characterize the rule combination in a mathematical way.

This paper is organized as follows. The FNN and its detailed operations are introduced in Section 2. A rule combination theorem and a practical search procedure will be proposed in Section 3. In Section 4, an example is given to illustrate the rule combination in the FNN. The final section is the conclusion.

2. Fuzzy neural network

The class of fuzzy inference system under consideration uses a singleton to represent the output fuzzy set of every fuzzy rule. Jou [3] has given a discrete equation to represent the numerical output by applying

* Corresponding author.

SSDI 0165-0114(95)00252-9

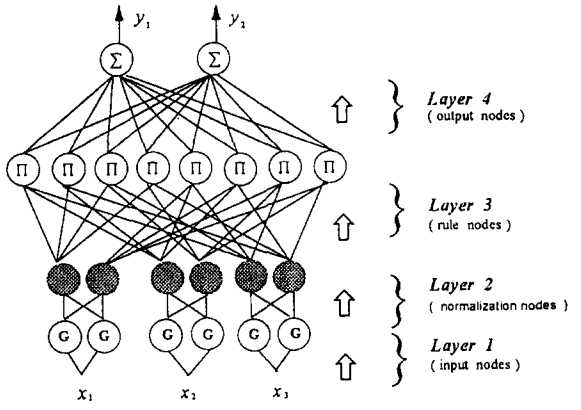

Fig. 1. The structure diagram of the FNN.

max-product composition and *centroid defuzzification* procedure to this fuzzy inference system. We quote this equation in the following.

$$y = \frac{\sum_{j=1}^{m} \beta^j (\prod_{i=1}^{n} \mu_{A_i^j}(x_i))}{\sum_{j=1}^{m} \prod_{i=1}^{n} \mu_{A_i^j}(x_i)}, \tag{1}$$

where $x_i \in X_i$, $y \in Y$, X_i and Y are the universes of discourse of input and output, respectively. The n-array variable $x = (x_1, \ldots, x_n)$ denotes the input vector, and $\mu_{A_i^j}(\cdot)$ is the membership function. β^j is the jth consequence singleton of output fuzzy set, m is the number of rules. We use two kinds of notation to represent the fuzzy set of X_i, throughout this paper. They are: (1) A_i^j denotes the fuzzy set of X_i associated with jth rule and (2) A_{ij} denotes the jth fuzzy set of X_i. In this special form we can describe the jth fuzzy rule as follows:

If x_1 is A_1^j and x_2 is A_2^j
and ... and x_n is A_n^j then y is β^j.

Based on (1), a four-layered fuzzy neural network (FNN) has been constructed. The structure diagram of the FNN is shown in Fig. 1. Notice that layer 2 (normalization layer) is not inherent in (1). For simplicity, we show an FNN with 3 inputs and 2 outputs in Fig. 1. In the following, we will explain the physical meaning and the node functions for each layer.

2.1. Layered operation of FNN

Layer 1: linguistic term layer

This layer uses Gaussian function as membership function, so these term nodes map input x_i onto their membership degree $\mu_{A_{ij}}(x_i)$ by using the jth term node of x_i, i.e.,

$$\mu_{A_{ij}}(x_i) = \exp\left(- \left(\frac{x_i - m_{ij}}{\sigma_{ij}} \right)^2 \right), \tag{2}$$

where m_{ij} and σ_{ij} denote mean (center) and variance (width) with respect to A_{ij}.

Layer 2: normalization layer

This layer performs normalization procedure for layer 1 (i.e. the sum of the whole term sets for each input is unity). Notice that no weight is adjusted here, i.e.,

$$\mu'_{A_{ij}}(x_i) = \frac{\mu_{A_{ij}}(x_i)}{\sum_{j=1}^{n_i} \mu_{A_{ij}}(x_i)}, \tag{3}$$

where n_i is the number of term nodes (or fuzzy partitions) of x_i. $\mu'_{A_{ij}}(x_i)$ is the normalized output of $\mu_{A_{ij}}(x_i)$. This layer is important for rule combination, as will be explained in Section 3.

Layer 3: rule layer

This layer implements the related link for preconditions (term node) and consequence (output node). There is still no weight adjusted at this layer. The jth output of layer 3 is

$$o_j^3 = \prod_{i=1}^{n} \mu'_{A_{ij}}(x_i). \tag{4}$$

Layer 4: output layer

This layer performs the centroid defuzzification to get numerical output. We can directly use (1) by replacing $\mu_{A_i^j}(x_i)$ with $\mu'_{A_i^j}(x_i)$. Then the output is

$$y = \sum_{j=1}^{m} \beta^j \prod_{i=1}^{n} \mu'_{A_i^j}(x_i). \tag{5}$$

Here the denominator has been eliminated since the denominator is

$$\sum_{j=1}^{m} \prod_{i=1}^{n} \mu'_{A_i^j}(x_i) = \prod_{i=1}^{n} \sum_{j=1}^{ni} \mu'_{A_{ij}}(x_i) = 1, \quad \forall x_i \in X_i. \tag{6}$$

The adjusted parameters in the FNN can be divided into two categories based on IF (premise) part and THEN (consequence) part in the fuzzy rules. A gradient descent based BP algorithm [6] is employed to adjust the FNN's parameters. In the premise part, the parameters are mean and width of Gaussian function. For initializing these terms, we use normal fuzzy sets since our main purpose lies in the studies of rule combination. In the consequence part, the parameters are output singletons. These singletons are initialized with small random values.

3. Rule combination

Once the supervised learning is finished, we know only that more rules result in better performance. We have no way of ensuring, however, that all of the rules are necessary. On the other hand, we also know that it may not be necessary to use all of the preconditions (A_{ij}') to determine fuzzy if–then rules in a particular case (i.e., some inputs have no effect on the specific rules introduced). This situation occurs frequently. For example, if a continuous function is of the form $f(x_1,x_2) = x_1 e^{-x_2} + 1$, then if $x_1 = 0$, we have $f = 1$, $\forall x_2$; f is not affected by x_2 in this case. x_2 clearly is a redundant input. We now introduce the following definition of rule combination:

Rule Combination is a procedure (a) for eliminating redundant preconditions of the fuzzy rules and (b) for combining the fuzzy rules into one single, equivalent rule.

The conditions for applying rule combination has been explored in [5] by using three conditions. These conditions for rule combination can be used to the proposed FNN with a slight modification. If some rule nodes satisfy the following conditions, then these rules can be combined into one rule.

(i) *These rule nodes have exactly the same consequence weights.* But in practical simulation results, we have found that rules seldom have exactly the same consequence weights without any approximation.

(ii) *The rule nodes satisfying condition* (i) *share some common preconditions* (i.e. the rule nodes are associated with the same term nodes).

(iii) *The union of the remaining preconditions support a whole set of term nodes to their associating inputs.*

An example of rule combination is illustrated in Fig. 2. For simplicity, Fig. 2 shows only the normalization layer, rule layer, and output layer of the FNN. In Fig. 2, four rules have the same consequence weight β^*. This satisfies condition (i). The first term set of x_1 is the common precondition of these four rules, thus condition (ii) is satisfied. The remaining preconditions are the term sets of x_2 and x_3. The question now is whether the remaining preconditions comprise all of the term sets of the associated inputs. The whole term sets of x_2 are A_{21} and A_{22}. From Fig. 2, we can see that the condition (iii) is satisfied for the input x_2. Similarly, the whole term sets of x_3 are A_{31} and A_{32}. From Fig. 2, we see that the condition (iii) is also satisfied for the input x_3. Now we can use only the common precondition to form a new rule. All of the term sets of x_2 and x_3 are redundant preconditions of these four rules.

In order to treat a mathematical equation as an *expression*, we denote a_{ij} and a_i^j as the output variables of layer 2 associated with $\mu'_{A_{ij}}(x_i)$ and $\mu'_{A_i^j}(x_i)$, respectively. By using (5) and (6), we can apply simple algebra to illustrate the example as shown in Fig. 2:

$$\begin{aligned} y &= \beta^*\{a_{11}a_{21}a_{31} + a_{11}a_{21}a_{32} \\ &\quad + a_{11}a_{22}a_{31} + a_{11}a_{22}a_{32}\} \\ &= \beta^* a_{11}(\underbrace{a_{21} + a_{22}}_{1})(\underbrace{a_{31} + a_{32}}_{1}) = \beta^* a_{11} \end{aligned} \tag{7}$$

Using conditions (i)–(iii), we will state a theorem for rule combination. Before going further, however, we need to introduce the following notation:

$$R_c' = \{k \mid \beta^k = \beta^*,\ k = 1, \ldots, m'\}, \tag{8}$$

$$I^+ = \{i \mid A_i^k = A_i^l;\ \forall k, l \in R_c', i = 1, \ldots, n\}, \tag{9}$$

$$I^- = \{i \mid A_i^k \neq A_i^l;\ \text{for some } k, l \in R_c'\}, \tag{10}$$

where β^k denotes the kth output singleton and β^* is a constant. In fact, (8) is just the condition (i). Eq. (9) indicates where the common preconditions exist; this has the same meaning as the condition (ii). The following *rule combination theorem* will complete the conditions (i)–(iii).

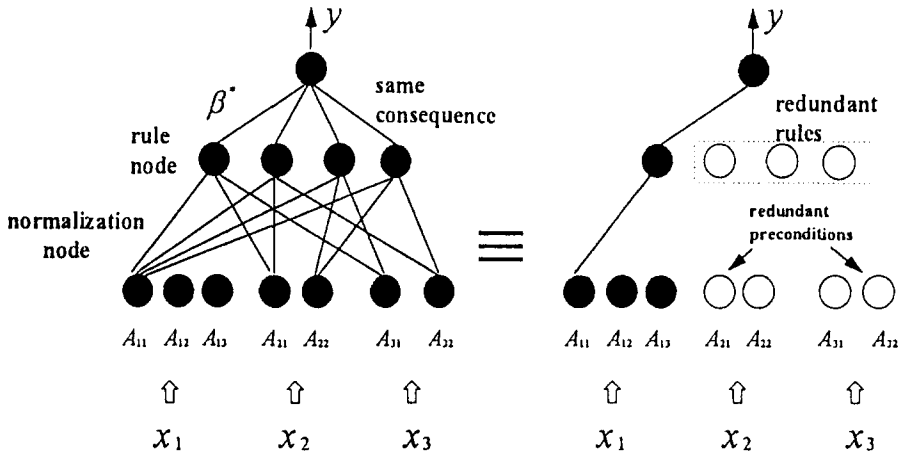

Fig. 2. An example illustrating rule combination.

Rule combination theorem. *If the expression*

$$\sum_{k\in R_c'}\prod_{i\in I}^{n} a_i^k \tag{11}$$

can be factorized as

$$\left(\prod_{i\in I^+} a_i^k\right)\left(\prod_{i\in I^-}\sum_{j=1}^{n_i} a_{ij}\right), \tag{12}$$

then (a) *the A_{ij}'s are redundant preconditions, for $i \in I^-$;* (b) *the rules with index k can be combined into one rule which uses only A_i^k as preconditions, for $i \in I^+$, $k \in R_c'$.*

Proof. From (5), we obtain

$$\begin{aligned} y &= \sum_{k\notin R_c'}\beta^k\prod_{i=1}^{n} a_i^k + \sum_{k\in R_c'}\beta^*\prod_{i=1}^{n} a_i^k \\ &= \sum_{k\notin R_c'}\beta^k\prod_{i=1}^{n} a_i^k + \beta^*\sum_{k\in R_c'}\prod_{i=1}^{n} a_i^k \\ &= \sum_{k\notin R_c'}\beta^k\prod_{i=1}^{n} a_i^k + \beta^*\left(\prod_{i\in I^+} a_i^k\right)\left(\prod_{i\in I^-}\sum_{j=1}^{n_i} a_{ij}\right) \\ &= \sum_{k\notin R_c'}\beta^k\prod_{i=1}^{n} a_i^k + \beta^*\left(\prod_{i\in I^+} a_i^k\right), \end{aligned} \tag{13}$$

where $\sum_{j=1}^{n_i} a_{ij} = 1$

From (9) and (10) we know that $I^+ \cap I^- = \emptyset$, and for $i \in I^-$, a_{ij}'s do not emerge in (13), so A_{ij}, $\forall j$, are all redundant preconditions. From (13) we also see that these rules with rule index $k \in R_c'$ are combined into one rule. □

From (12) we know that the number of all reduced rules is $(\prod_{i\in I^-} n_i) - 1$, when one rule combination is taken. A heuristic method for putting rule combination theorem into practice is proposed in the following.

If we can find the set R_c' then the rule combination procedure can be applied by using the following methods.

Rule combination search procedure

Initialization: Define $I_R = \{1, \ldots, n\}$.

Step 1: Suppose i^* is a candidate for being a redundant input associated with these rules.

Step 2: Search from $j = 1$ to $j = n_{i^*}$. If all of the terms A_{i^*j} can be found, then go to step 4. Else go to step 3.

Step 3: $I_R = I_R - \{i^*\}$. If $I_R = \emptyset$ then stop, else go to step 5.

Step 4: Search from $j = 1$ to $j = n_{i^*}$ to find a common factor. If a common factor is found, then eliminate the terms A_{i^*j} from the associated rules. Repeat this step, until no further common factors are found, go to step 3.

Step 5: Set a new candidate i^* which has not yet been searched and go to step 2.

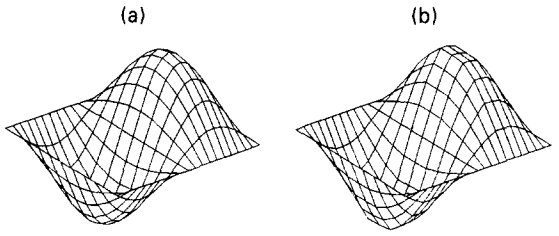

Fig. 3. The desired (a) and reconstructed (b) surface of $f(x_1,x_2)$.

Table 1
Initial and final parameters of the membership functions.

Term sets	Initial parameters		Final paremeters	
	Mean	Variance	Mean	Variance
A_{11}	−1.000	0.127	−0.973	0.170
A_{12}	−0.667	0.127	−0.651	0.114
A_{13}	−0.333	0.127	−0.352	0.114
A_{14}	−0.000	0.127	0.004	0.202
A_{15}	0.333	0.127	0.361	0.124
A_{16}	0.667	0.127	0.674	0.107
A_{17}	1.000	0.127	0.990	0.144
A_{21}	0.000	0.095	−0.000	0.092
A_{22}	0.250	0.095	0.250	0.100
A_{23}	0.500	0.095	0.510	0.095
A_{24}	0.750	0.095	0.752	0.098
A_{25}	1.000	0.095	1.010	0.095

4. Numerical example

The following example of continuous functions is presented to illustrate the proposed rule combination search procedure.

4.1. Example

$$f(x_1,x_2) = \sin(\pi x_1)\sin(\pi x_2),$$

for $-1 \leqslant x_1 \leqslant 1$ and $0 \leqslant x_2 \leqslant 1$.

The initial structure of the FNN uses 7 and 5 term nodes for x_1 and x_2 respectively, i.e. in this case we have 7×5 initial rules. The optimal choice of the number of term nodes is still a difficult problem. In our simulation, we tried several cases and found an acceptable choice. Here we use 231 training patterns collected from f. The input/output relation of f is shown in Fig. 3(a). The fuzzy sets for these linguistic term nodes are normally and uniformly initialized. The back-propagation algorithm with learning rate 0.01 is used to train the FNN with 300 epochs. The initial and final parameters of the FNN are shown in Table 1. The rule combination search procedure then is applied and the rule number is reduced to 17. At last, we eliminate some petit rules (i.e., their consequence weights are very small). The final rules number is reduced to 12. Fig. 3(b) shows the reconstructed' surface using the FNN.

R^1 : If x_1 is A_{12} and x_2 is A_{22} then $f(x_1,x_2)$ is -0.690.

R^2 : If x_1 is A_{13} and x_2 is A_{22} then $f(x_1,x_2)$ is -0.732.

R^3 : If x_1 is A_{15} and x_2 is A_{22} then $f(x_1,x_2)$ is 0.754.

R^4 : If x_1 is A_{16} and x_2 is A_{22} then $f(x_1,x_2)$ is 0.552.

R^5 : If x_1 is A_{12} and x_2 is A_{23} then $f(x_1,x_2)$ is -0.976.

R^6 : If x_1 is A_{13} and x_2 is A_{23} then $f(x_1,x_2)$ is -1.035.

R^7 : If x_1 is A_{15} and x_2 is A_{23} then $f(x_1,x_2)$ is 1.006.

R^8 : If x_1 is A_{16} and x_2 is A_{23} then $f(x_1,x_2)$ is 0.780.

R^9 : If x_1 is A_{12} and x_2 is A_{24} then $f(x_1,x_2)$ is -0.690.

R^{10} : If x_1 is A_{13} and x_2 is A_{24} then $f(x_1,x_2)$ is -0.732.

R^{11} : If x_1 is A_{15} and x_2 is A_{24} then $f(x_1,x_2)$ is 0.754.

R^{12} : If x_1 is A_{16} and x_2 is A_{24} then $f(x_1,x_2)$ is 0.552.

5. Conclusion

We have described the architecture of the FNN. By employing the supervised learning procedure, the proposed architecture can refine fuzzy if–then rules obtained from human experts to describe the input–output behavior of a complex system. However, if human expertise is not available, we can still set up reasonable process to generate a set of fuzzy if-then rules to approximate a desired data set, as shown in the simulation example of function approximation. Also, by employing the process of the rule combination, the FNN greatly reduce the cost of the structures. To make the proposed rule combination more efficient and practical, we could consider incorporating an approximation method into the search procedure. With slight modification, the proposed rule combination procedure can also be extended to an MIMO case. Though, we did not pursue along these directions, however, they are expected for further understanding.

References

[1] S. Horikawa, T. Furuhashi and Y. Uchikawa, On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm, *IEEE Trans. Neural Networks* **3**(5) (1992) 801–806.

[2] J.S. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Trans. Systems Man Cybernet.* **23**(3) (1993) 665–684.

[3] C.C. Jou, On the mapping capability of fuzzy inference system, *Proc. Internat. Joint. Conf. on Neural Networks*, Baltimore, ML (1992) 708–713.

[4] J.M. Keller, R.R. Yager and H. Tahani, Neural network implementation of fuzzy logic, *Fuzzy Sets and Systems* **45** (1992) 1–12.

[5] C.T. Lin and C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.* **40**(12) (1991) 1320–1336.

[6] D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing*, Vol. 1 (MIT Press, Cambridge, MA, 1986).

[7] M. Sugeno and T. Yasukawa, A fuzzy-logical-based approach to qualitative modeling, *IEEE Trans. Fuzzy Systems* **1**(1) (1993) 7–31.