Genetic-Fuzzy Modeling on High Dimensional Spaces

Joon-Min Gil[1] and Seong-Hoon Lee[2]

[1] Dept. of Computer Science Education, Catholic University of Daegu
330 Geumnak, Hayang-eup, Gyeongsan-si, Gyeongbuk 712-702, Korea
jmgil@cu.ac.kr
[2] Dept. of Computer Science, Cheonan University
115 Anseo-dong, Cheonan 330-180, Korea
shlee@cheonan.ac.kr

Abstract. In this paper, in order to reduce the explosive increase of the search space as the input dimension grows, we present a new representation method for the structure of fuzzy rules, *a graph structured fuzzy system*. The graph structured fuzzy system can flexibly cope with the increase of the input space by selecting these fuzzy rules that significantly affects the input space among the whole set of fuzzy rules. To obtain the optimal structure and parameters of fuzzy systems, an approach to the automatic design of fuzzy systems based on L-systems is also proposed. The proposed method can efficiently construct fuzzy rules without any need for user interaction by using the rewriting mechanism of L-systems.

1 Introduction

When defining an optimal fuzzy system for practical problems, we know intuitively that the whole set of fuzzy rules is impractical, in particular, if the problems are complex; i.e., a fuzzy system with the whole set of fuzzy rules may have a possibility to be included inadequately defined fuzzy rules or conflicted fuzzy rules which are very difficult to identify complex systems. Accordingly, the form of a lookup table expressing the whole set of fuzzy rules can not represent an optimal subset of fuzzy rules. To alleviate this problem, as the first objective of this paper, we propose a new representation method for the structure of fuzzy rules, *a graph structured fuzzy system*. The graph structured fuzzy system can flexibly cope with the increase of the input dimension by selecting these fuzzy rules that significantly affects the input space among the whole set of fuzzy rules.

It is pointed out that an optimal fuzzy system can be extracted from the definition of both the nodes and edges in a graph structured fuzzy system. This is because a good graph structure can generate enough fuzzy rules to precisely represent input-output relation. Genetic algorithms can be utilized to find such the graph structure. However, previous works have mainly focused on the method which directly encodes parameters in fuzzy systems into chromosomes [1,2]. Considering practical problems with high-dimensional input space, as the number of the parameters increase, the complicated search space may still arise from the direct encoding of these parameters. To overcome the problem associated with

B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): KES 2006, Part I, LNAI 4251, pp. 1147–1154, 2006.

the encoding, as the second objective of this paper, we propose the automatic generation method of a fuzzy system, which is is based on L-systems that can define complex objects by successively replacing parts of a simple object using a set of rewriting production rules. The rewriting production rules of L-systems, which can derive the optimal set of fuzzy rules, are encoded into chromosomes in our genetic algorithms. Therefore, our design method can find the optimal structure of a fuzzy system with the reduced search space. Moreover, it can reduce computational requirements for identifying a fuzzy system because the generated system has only the essential fuzzy rules, which are eliminated the inadequate fuzzy terms through evolutionary search.

This paper is organized as follows. In Sec. 2, we provides a brief description of a simplified fuzzy model. Sec. 3 describes a design method of fuzzy systems for solving the explosive increase of search space under high-dimensional input spaces. In this section, we introduce a graph structured fuzzy system. The generation method of the fuzzy system based on L-systems is given in Sec. 4. In Sec. 5, the genetic algorithm used for letting the rewriting production rules of L-systems to be optimized is presented. Sec. 6 shows the simulation with a time-series prediction problem. Finally, our conclusion is given in Sec. 7.

2 Simplified Fuzzy Model

This paper uses the simplified fuzzy model [3] described by the following fuzzy IF-THEN rules: `IF` x_1 `is` A_1^i `AND`, $\ldots$, `AND`, x_d `is` A_d^i, `THEN` y `is` w_i. Here, A_j^i is a fuzzy membership function for the input variable x_j in the ith fuzzy rule ($i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, d$); w_i is a real value for the output variable y; n and d are the number of fuzzy rules and input variables, respectively.

Given the real-valued input vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]$, the real-valued output of the fuzzy model is inferred as follows:

$$f(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \mu_i \cdot w_i}{\sum_{i=1}^{n} \mu_i}, \ \mu_i = \prod_{j=1}^{d} \exp\left(-\frac{1}{2} \cdot \left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right) \tag{1}$$

where, μ_i implies the overall truth value of the premise of the ith implication for the input; c_j^i and σ_j^i are the central value and the width of a gaussian fuzzy membership function, respectively.

In order to design a sophisticated fuzzy system, three parameters c_j^i, σ_j^i, and ω_i need to be adjusted by an identification algorithm. This paper uses a self-learning method by the delta rules [3]. The structure identification of the simplified fuzzy model is introduced in next section.

3 Graph Structured Fuzzy System

In the design phase of fuzzy systems, it is useful to select these fuzzy rules that significantly affects an input space among the whole set of fuzzy rules [2]. Let

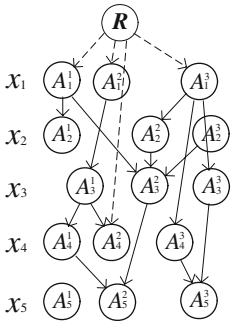

Fig. 1. Graph structure

IF x_1 is A_1^1 AND x_2 is A_2^1 THEN y is w_1
IF x_1 is A_1^1 AND x_3 is A_3^2 AND x_5 is A_5^2 THEN y is w_2
IF x_1 is A_1^1 AND x_3 is A_3^1 AND x_4 is A_4^1 AND x_5 is A_5^2 THEN y is w_3
IF x_1 is A_1^2 AND x_3 is A_3^1 AND x_4 is A_4^2 THEN y is w_4
IF x_4 is A_4^2 THEN y is w_5
IF x_1 is A_1^3 AND x_2 is A_2^2 AND x_3 is A_3^2 AND x_5 is A_5^2 THEN y is w_6
IF x_1 is A_1^3 AND x_4 is A_4^3 AND x_5 is A_5^3 THEN y is w_7
IF x_1 is A_1^3 AND x_3 is A_3^3 AND x_5 is A_5^3 THEN y is w_8

Fig. 2. Fuzzy rules generated by a graph structure

the input space be the cube of r dimension $[-L, L]^r$ and each input domain be partitioned into N fuzzy terms. Then, all the input spaces are partitioned into N^r. In the conventional fuzzy systems, the number of fuzzy rules is N^r. Accordingly, the total number of parameters are $a \cdot r \cdot N + b \cdot N^r$, where a is the number of parameters for fuzzy membership functions and b is the number of parameters for the real parts of fuzzy rules. This means that the fuzzy systems need the order of N^r fuzzy rules in order to approximate a real system. As a result, the fuzzy systems suffer from an exponential rule explosion as the input dimension r grows. As proven in [3,4], there exists a fuzzy system that can approximate a practical problem. However, it is not easy to find such the fuzzy system, in particular, if the problem is complex.

This paper defines fuzzy rules as a graph structure. In the graph structure, nodes consist of a root one and verbal ones. Each verbal node represents a fuzzy membership function for an input variable. The relation between two nodes is expressed as directed edges. Fig. 1 shows an arbitrary graph structure when 5 input values and 3 membership functions are used. In this figure, R and A_j^i represent a root node and a verbal one, respectively. A dashed line represents edges between the root node and a verbal one. The edges between two verbal nodes are depicted as a solid line.

The conversion of the graph structure into fuzzy rules is to search all the paths from the root node to the verbal node which has not any more branch. Fig. 2 shows the fuzzy rules generated by the graph structure shown in Fig. 1. For example, the path $R \dashrightarrow A_1^1 \rightarrow A_2^1$ in Fig. 1 is interpreted as the fuzzy rule whose premise part includes the verbal nodes A_1^1 and A_2^1 but does not exist the input variables x_3, x_4, and x_5. In this way, total 8 fuzzy rules are generated.

Fig. 2 illustrates that fuzzy rules do not necessarily have to involve all the input spaces; i.e., the structure of fuzzy rules defined in this figure can eliminate unnecessary input variables and membership functions from the whole of the input spaces. It should be noted that fuzzy rules can be optimized by the selective combination of membership functions for each input variable.

4 Generation of Fuzzy Systems by L-Systems

In this section, we describe the design method of fuzzy systems based on L-systems, termed DOL-systems. L-systems are a mathematical formalism proposed by biologist Aristid Lindenmayer in 1968 as a foundation for an axiomatic theory of biological development [5]. The principle notion of L-systems is rewriting, where the basic idea is to define complex objects by successively replacing parts of a simple object using a set of rewriting production rules.

It is important to define the optimal nodes and edges of a graph structured fuzzy system in order to obtain an optimal graph structure. This paper defines a set of rewriting production rules from which the optimal nodes and edges of the fuzzy system can be obtained. Let us consider a definition of the nodes that represent fuzzy membership functions. The central value of membership functions on the input domain $[L, R]$ is defined as follows: $c_i = L + i \times \frac{(R-L)}{C}$. Here, C is a precision for the input domain $[L, R]$ $(i = 1, 2, \ldots, C)$.

The edges representing the structure of fuzzy rules are extracted from L-systems defined in the below. The L-systems automatically generate the nodes and edges of a graph structure by the interpretation of the command that can change a graph state successively. The state of a graph is defined by (x, y), where x is the index of an input variable and y is the index of a membership function. Then, the graph responds to the commands defined by

Fm Move forward a step of length m. The state of the graph changes to (x', y'), where $x' = x + m$ and $y' = y$. A edge between two verbal nodes (x, y) and (x', y') is made.

fm Move forward a step of length m. The state of the graph changes as above. This command is applied to only between the root node R and a verbal node (x', y'). A edge between two nodes is also made.

$+n$ Move right a step of length n. The state of the graph changes to (x', y') without making a edge, where $x' = x$ and $y' = y + n$.

$-n$ Move left a step of length n. The state of the graph changes to (x', y') without making a edge, where $x' = x$ and $y' = y - n$.

In order to represent the graph structure efficiently, the alphabet of L-systems is extended with two new symbols, '[' and ']'. They are interpreted by the graph as follows:

[Push the current state of the graph onto a pushdown stack.

] Pop a state from the stack and replace it with the current state of the graph.

Given a string v and an initial state of the graph (x_0, y_0), a graph structure is generated by successively rewriting the graph produced by the interpretation of rewriting production rules. Here, v is composed of an initial symbol S and a set of rewriting production rules $\{p_1, p_2, \ldots\}$. The graph structure shown in Fig. 1 can be obtained by interpreting the strings of the following L-system: $w : S$, $S \rightarrow p_1|p_2|p_3|p_4$, $p_1 \rightarrow +1f1[+0F1]+1F2+0F2$, $p_2 \rightarrow +2f1-1F2[+0F1+1F1]+1F1$, $p_3 \rightarrow +2f4$, $p_4 \rightarrow +3f1[-1F1 + 0F1 + 0F2][+0F3 + 0F1] + 0F2 + 0F2$.

5 Optimization of Production Rules

This section describes genetic algorithms for optimizing the rewriting production rules by which an optimal graph structure can be obtained.

5.1 Encoding

Since an initial symbol is always S, only the rewriting production rules are encoded into a chromosome. The chromosome is subdivided into three subchromosomes: one is for commands, the others are for push and pop operations in a push-down stack. For example, the rewriting production rule p_4 is encoded as follows: $\{\{+3f1-1F1+0F1+0F2+0F3+0F1+0F2+0F2\}, \{010010000\}, \{000010100\}\}$. Here, commands are encoded into first part of the chromosome. For push and pop symbols, the integers which are correspondent with the number of '[' and ']' are encoded into second and third part of the chromosome, respectively.

5.2 Genetic Operations

Since rewriting productions rules have a variable length, each chromosome has also a variable length. Accordingly, to allow a proper derivation and interpretation of genotypes, genetic operations must produce offspring with a valid graph structure. With this consideration, four main operators are designed: two kinds of crossovers and two kinds of mutations.

- **Crossover:** This operator introduces information exchange between two chromosomes to create offsprings. According to the substrings to be exchanged, two kinds of crossovers are designed.
  a. Production Crossover: This is applied to the unit of rewriting production rules. Since a production rule can be interpreted to several fuzzy rules, the information exchanged between two parents is also fuzzy rules. Uniform crossover is used for this crossover.
  b. Symbol Crossover: This is inspired by the Genetic Programming crossover [6]. Koza's Lisp subtrees can be considered analogous to correctly substrings within a L-system. This crossover can search finer fuzzy rules by exchanging the unit of commands.
- **Mutation:** This operator introduces random variations in rewriting production rules.
  a. Symbol Mutation: For the symbols '+' and '−', this mutation changes '+' into '−' or vice verse with a given probability. m and n each is replaced by the value of the randomly selected integer within the given range of values. The value of the genes represented for push and pop operations is also replaced by the value of a randomly selected integer.
  b. Block Mutation: A randomly selected block in a rewriting production rule is replaced by the random string which is syntactically correct.

5.3 Fitness Function

The graph structure generated by the interpretation of L-systems is applied to the evaluation of a chromosome. Accordingly, a fitness function is based on the phenotype representing a graph structure. Generally, as a proper criterion for the verification of fuzzy systems, a learning error has been used [1,2]. Also, the number of fuzzy rules can be used as the design objective of fuzzy systems in order to minimize computational complexity. The fitness function with these objectives is defined by $f(s_i) = \frac{1}{\alpha_e \cdot e + \alpha_r \cdot r}$. Here, $f(s_i)$ is a fitness value for the ith chromosome s_i; e and r denote a learning error and the number of fuzzy rules, respectively; α_e and α_r are nonnegative weights for a learning error and the number of fuzzy rules, respectively.

6 Simulation and Results

To demonstrate the efficiency of the proposed method, we apply the proposed one to the time series prediction problem. The time series used in this paper is generated by the chaotic Mackey-Glass differential delay equation [3] defined as

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{2}$$

The prediction problem is to predict the value at some point in the future $t+P$, $x(t+P)$, using the past values of the time series up to time t, $\{x(t), x(t-\Delta t), \ldots, x(t-(n-1)\Delta t)\}$. According to the selection of Δt and n for P, the number of input dimensions and the property of the problem are determined. In our simulation, the values $P = \Delta t = 6$ and $n = 6$ are used. We use the fourth order Runge-Kutta method to find the numerical solution for (2). Also, the time step used in the simulation is 0.1, initial condition $x(0)$ is 0.8, and τ is 30. In this way, we extract 1000 data from $130 \leq t \leq 1129$. First 500 pairs are used as training data and the remaining 500 pairs used as checking data. The membership functions for all the input variables are defined in the range $[0.15, 1.35]$ and each input domain are partitioned into nine parts; i.e., $C = 9$. We assign 10 rewriting production rules to each chromosome. The probabilities of crossover and mutation are 0.3 and 0.1, respectively. In order to compare the proposed method with other design one, we use the normalized mean squared error (NMSE) defined as the root mean squared error divided by the data standard deviation.

After 100 generations, our genetic algorithm finds the L-system shown in Fig. 3. In the initial phase of the genetic algorithm, 10 rewriting production rules are randomly encoded into chromosomes, but finally only 8 rewriting production rules are used for constructing the graph structure shown in Fig. 4. Since remaining two rewriting production rules generate an invalid graph state, they are excluded from the construction of the graph structure.

By the graph interpretation of the rewriting production rules, we obtained 15 membership functions and 15 fuzzy rules. Initially, each input space is partitioned into nine parts and therefore total 54 membership functions can be used

$w : S,$
$S \rightarrow p_1 \mid p_2 \mid p_3 \mid p_4 \mid p_5 \mid p_6 \mid p_7 \mid p_8,$
$p_1 \rightarrow +5f4[-2F1] - 0F1[-3F1][-1F1],$
$p_2 \rightarrow [+4f2] + 4f1[-3f2],$
$p_3 \rightarrow +3f6,$
$p_4 \rightarrow +5f5[-3F1][-1F1] + 2F1,$
$p_5 \rightarrow +1f1 + 0F1,$
$p_6 \rightarrow +8f1,$
$p_7 \rightarrow +8f5[-0F1],$
$p_8 \rightarrow +4f6.$

Fig. 3. L-system obtained by evolutionary search

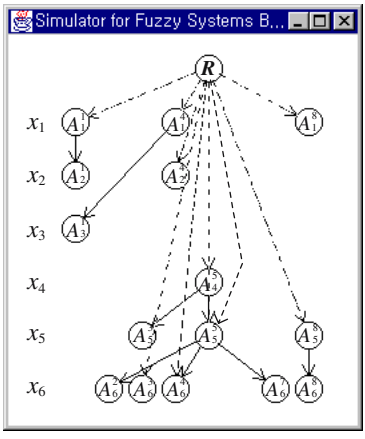

Fig. 4. Graph structure generated by the interpretation of L-system

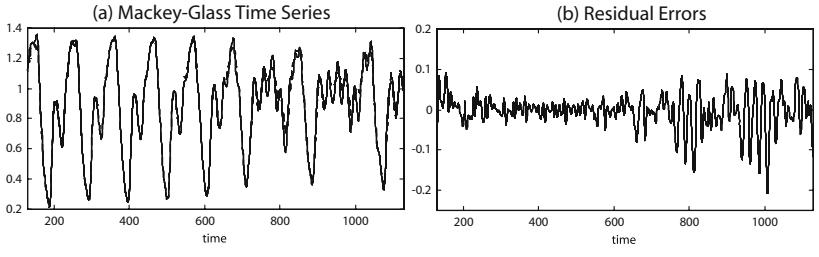

Fig. 5. Prediction results

as verbal nodes. In case of the lookup table structure of fuzzy rules, entire 54 membership functions are used in fuzzy systems without any elimination for the illegal definition of membership functions. Such fuzzy systems will have difficulty in partitioning input domains correctly for a given problem. However, our method selects only 15 membership functions which significantly affect the input space. The fuzzy rules constructed by all the combinations of 15 fuzzy membership functions would be 90 ($3 \times 2 \times 1 \times 1 \times 3 \times 5$), but our method produces only 15 fuzzy rules by the selective combination of the membership functions; i.e., when defining fuzzy rules, it can use only the membership functions that significantly affect the input space through evolutionary search. It should be noted that the fuzzy systems based on L-systems can generate only the fuzzy rules that can correctly express input-output relation for a given problem.

Fig. 5(a) shows the real curve of time series prediction problem and the predicted curve by our method. In this figure, a dashed line represents the real curve and a solid line the predicted curve. Two curves are visually indistinguishable. Fig. 5(b) shows the residual errors that indicate the difference of the real values from the predicted values at each time. Although the time series used

Table 1. Comparison of prediction results

		Conventional method	Proposed method
Fuzzy rules		29	15
NMSE	training	0.0509	0.0385
	checking	0.1637	0.1164

for checking data are more complex than training data, it is obvious that our method can well predict the real curve for checking data as well as for training data. Table 1 compares the results of our method with those of the conventional one where a graph structure is directly encoded into chromosomes. It shows that our method can generate much less NMSE than the conventional one, even though a few fuzzy rules are used.

7 Conclusion

An approach to the automatic design of fuzzy systems by L-systems was proposed in this paper. In order to reduce the explosive increase of the search space as the input dimension grows, both membership functions and fuzzy rules are constructed by using the rewriting mechanism of L-systems. The generated fuzzy system has only the essential membership functions and fuzzy rules, which are obtained by eliminating inadequately defined fuzzy rules and unnecessary membership functions from all the input spaces through evolutionary search.

The simulation showed that our method can generate much less NMSE than the conventional one, even though a few fuzzy rules are used. From the obtained results, we concluded that our method can consistently reduce the computational complexity for designing fuzzy systems by efficiently reducing both the parameters of fuzzy systems and the search space.

We have evaluated with a specific time series prediction to show its feasibility on high-dimensional input spaces. To verify the applicability of our method in various areas, several simulations are being carried out.

References

1. T.C. Chin and X.M. Qi, "Genetic algorithms for learning the rule base of fuzzy logic controller," *Fuzzy Sets and Systems*, vol. 97, pp. 1-7, 1998.
2. K. Shimojima, T. Fukuda, and Y. Hasegawa, "Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm," *Fuzzy Sets and Systems*, vol. 71, pp. 295-309, 1995.
3. J.-R. Jang, C.-T. Sun, and E.M. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, 1997.
4. J. Buckley, "Sugeno type controllers are universal controller," *Fuzzy Sets and Systems*, vol. 53, pp. 299-303, 1993.
5. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1996.
6. M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, 1996.