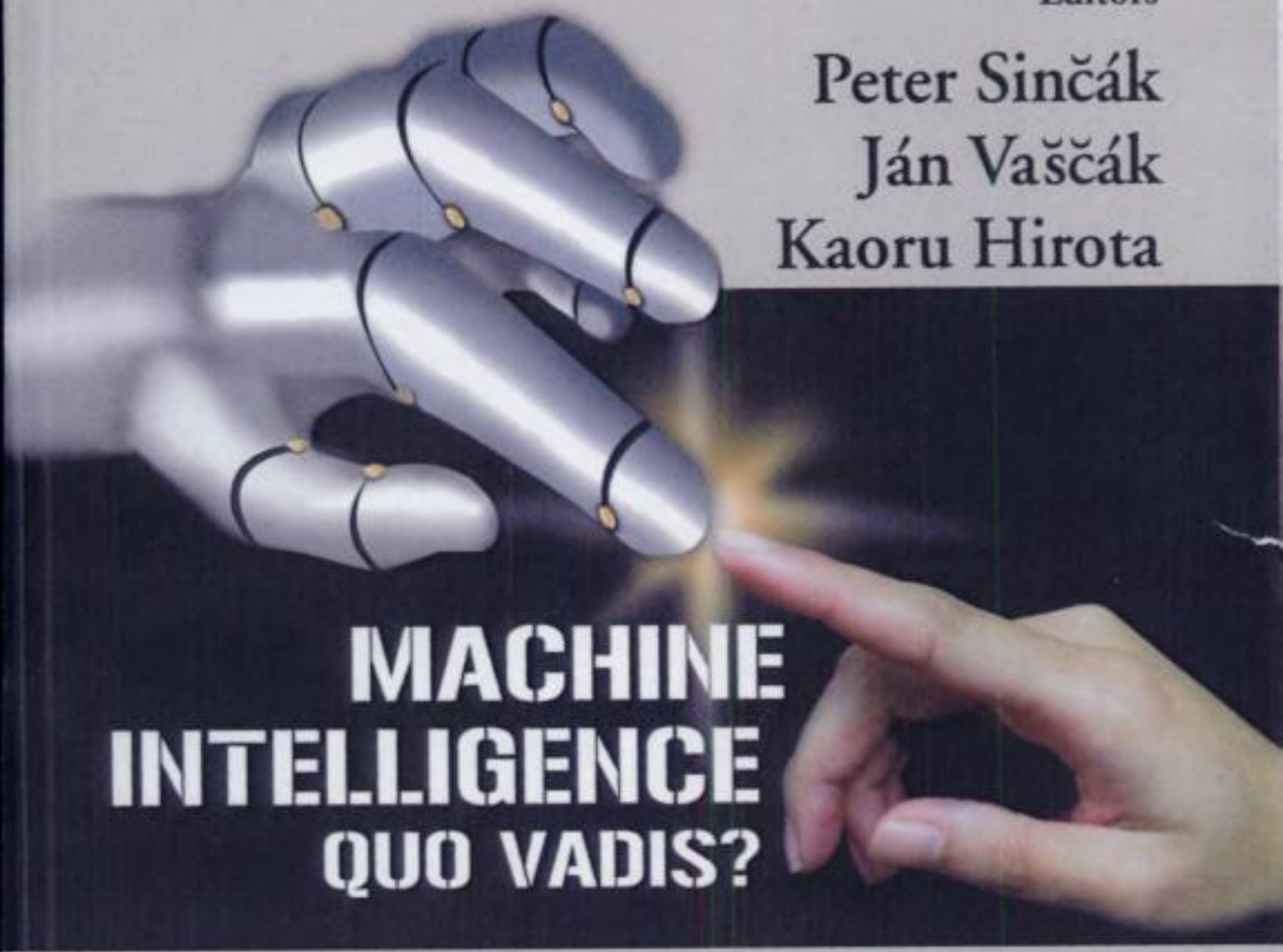
Advances in Fuzzy Systems — Applications and Theory – Vol. 21

Editors

Peter Sinčák
Ján Vaščák
Kaoru Hirota



MACHINE INTELLIGENCE
QUO VADIS?

World Scientific

FUZZY RULE EXTRACTION FROM INPUT/OUTPUT DATA

L. T. KÓCZY*, J. BOTZHEIM*, A. B. RUANO**, A. CHONG ***, T. D. GEDEON***

*Department of Telecommunication and Telematics,
Budapest University of Technology and Economics,
H-1117 Budapest, Pázmány P. sétány 1/d, Hungary
koczy@ttt.bme.hu
botzheim@alpha.ttt.bme.hu

** Department of Electronic Engineering and Computing,
Faculty of Sciences and Technology, University of Algarve,
8000 Faro, Portugal
aruano@ualg.pt

*** Department of Information Technology
Murdoch University, Australia
South Street, Murdoch 6150 WA
cchong@murdoch.edu.au
tgedeon@central.murdoch.edu.au

This paper discusses the question how the membership functions in a fuzzy rule based system can be extracted without human interference. There are several training algorithms, which have been developed initially for neural networks and can be adapted to fuzzy systems. Other algorithms for the extraction of fuzzy rules are inspired by biological evolution. In this paper one of the most successful neural networks training algorithm, the Levenberg-Marquardt algorithm, is discussed, and a very novel evolutionary method, the so-called "bacterial algorithm", are introduced. The class of membership functions investigated is restricted to the trapezoidal one as it is general enough for practical applications and is anyway the most widely used one. The method can be easily extended to arbitrary piecewise linear functions as well. Apart from the neural networks and evolutional algorithms, fuzzy clustering has also been used for rule extraction. One of the clustering-based rule extraction algorithms that works on the projection of data is also reported in the paper.

Keywords: fuzzy systems, fuzzy rules extraction, Levenberg-Marquardt algorithm, bacterial algorithm.

1 Introduction

In the application of fuzzy systems to modelling and control one of the most important tasks is to find the optimal rule base. This might be given by a human expert or might be given a priori by the linguistic description of the modelled system. If, however, neither a suitable expert, nor the necessary linguistic descriptions are available, the system has to be designed by other methods based on numerical data. In training, the objective is to tune the membership functions in the

fuzzy system such that the system performs a desired mapping of input to output. The mapping is given by a set of examples of this function, the so-called pattern set. Each pattern pair p of the pattern set consists of an input activation vector $x^{(p)}$ and its target activation vector $t^{(p)}$. After training the membership functions, when an input activation $x^{(p)}$ is presented, the resulting output vector $y^{(p)}$ of the fuzzy system should equal the target vector $t^{(p)}$. The distance between the target and the actual output vector must be minimised for each pattern. There are several methods to minimise these distances. These methods can be adopted from the field of neural networks or from the evolution phenomenon of living beings. The paper is organised as follows. Section 2 describes the basics of fuzzy systems. The neural network algorithm is shown in the Section 3. In Section 4 the bacterial algorithm is described. In section 5, 6, 7, 8 and 9, we complete our discussion by introducing a fuzzy clustering based rule extraction technique. Section 10 is the conclusion of the paper.

2 Fuzzy systems

The theory of fuzzy logic was developed by Zadeh in the early 1960s. His theory was essentially the rediscovering the multivalued logic created by Lukasiewicz, however, with going much further in some application related aspects. In 1973 he pointed out that the new fuzzy concept could be excellently used for describing very complex problems with a system of fuzzy relations represented by a fuzzy rule base [1]. A fuzzy rule base contains fuzzy rules R_i:

Ri: IF (x1 is Ai1) AND (x2 is Ai2) AND ... AND (xn is Ain) THEN (y is Bi). (1)

where A_{ij} and B_i are fuzzy sets, x_j and y are fuzzy inputs and output. The meaning of the structure of a rule is the following:

IF Premise THEN Conclusion (2)

where the premise consists of antecedents linked by fuzzy *AND* operators. The Centre of Gravity (COG) defuzzification method is used here because it is general and easy to compute. This method calculates the crisp output by the sums of the centre of gravities of the conclusions. Thus, a fuzzy inference system can compute output y of an input vector x. The main purpose is to make the best solution possible for each input vector, therefore the optimum rule base need to be found.

3 The Levenberg-Marquardt algorithm

Our goal is to find the optimal rule base. This means that the distances between the targets and the corresponding output vector gives smaller error than in the case of another rule base. The error is measured by the following function:

$$E = \frac{1}{2}\sum_{p=1}^{P}(t^{(p)} - y^{(p)})^2 \tag{3}$$

where P is the number of patterns in the pattern set, $t^{(p)}$ is the p^{th} target vector, $y^{(p)}$ is the p^{th} output vector. The most used method to minimise (3) is the Error-Back-Propagation (BP) algorithm, [7] which is a steepest descent algorithm. A newer method is the Levenberg-Marquardt algorithm. Denoting the parameter vector by $\underline{z}$, and the Jacobean matrix by $\underline{\underline{J}}$:

$$\underline{\underline{J}}[k] = \frac{\partial y[\underline{z}[k]]}{\partial \underline{z}^T[k]} \tag{4}$$

$\underline{s}[k] = \underline{z}[k] - \underline{z}[k-1]$ the LM update, is given as the solution of

$$(\underline{\underline{J}}^T[k]\underline{\underline{J}}[k] + \alpha \underline{\underline{I}})\underline{s}[k] = -\underline{g}[k] = -\underline{\underline{J}}^T[k]\underline{e}[k] \tag{5}$$

In (5), α is a regularization parameter, which controls the both the search direction and the magnitude of the update. (5) can be recast as:

$$\underline{s}[k] = -\begin{bmatrix} \underline{\underline{J}}[k] \\ \sqrt{\alpha}\,\underline{\underline{I}} \end{bmatrix}^{+} \begin{bmatrix} \underline{e}[k] \\ \underline{0} \end{bmatrix} \tag{6}$$

The complexity of this operation is of $O(n^3)$, where n is the number of columns of $\underline{\underline{J}}$. If we apply this algorithm in fuzzy systems then the parameters z must be found. The structure of the fuzzy system is the following:
A grid in the input space is defined. Vectors of *knots* must be defined, one for each input dimension. These vectors determine the place of the membership functions. In each i^{th} axis $\lambda_{i,j}$ will be defined, where $j = 0,1,...,r_i$. They are arranged in such a way that

$$\lambda_{i,0} \le \lambda_{i,1} \le ... \le \lambda_{i,r_i} \tag{7}$$

$\lambda_{i,0}$ is the minimal and λ_{i,r_i} is the maximal value of the i^{th} input. The membership functions can be interpreted as in Fig.1 (it is a quasi Ruspini-partition, namely it is a Ruspini-partition between $\lambda_{i,1}$ and λ_{i,r_i-1}).

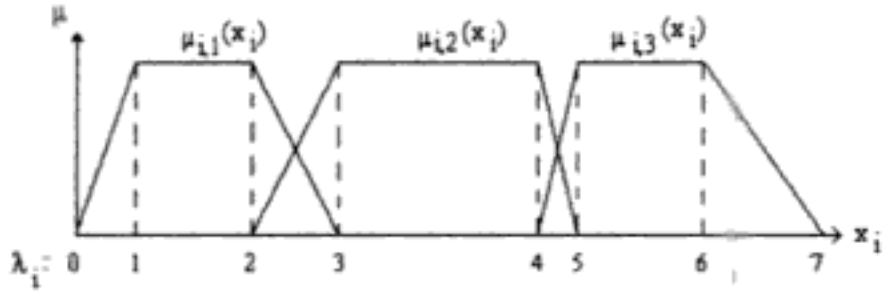


Fig.1. The positions of the membership functions

The j^{th} interval of the i^{th} input is denoted by $I_{i,j}$ and is defined as follows:

$$I_{i,j} = \begin{cases} [\lambda_{i,j-1} \quad \lambda_{i,j}) & for\ j = 1,\ldots, r_i - 1 \\ [\lambda_{i,j-1} \quad \lambda_{i,j}] & if\ j = r_i \end{cases} \tag{8}$$

The j^{th} membership function in the i^{th} dimension:

$$\mu_{i,j}(x_i) = \begin{cases} \dfrac{x_i - \lambda_{i,2j-2}}{\lambda_{i,2j-1} - \lambda_{i,2j-2}}, & if\ x_i \in I_{i,2j-1} \\ 1, & if\ x_i \in I_{i,2j} \\ \dfrac{\lambda_{i,2j+1} - x_i}{\lambda_{i,2j+1} - \lambda_{i,2j}}, & if\ x_i \in I_{i,2j+1} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

or in another form:

$$\mu_{i,j}(x_i) = \frac{x_i - \lambda_{i,2j-2}}{\lambda_{i,2j-1} - \lambda_{i,2j-2}} N_{i,2j-1}(x_i) + N_{i,2j}(x_i) + \frac{\lambda_{i,2j+1} - x_i}{\lambda_{i,2j+1} - \lambda_{i,2j}} N_{i,2j+1}(x_i) \tag{10}$$

where:

$$N_{i,j}(x) = \begin{cases} 1, & if\ x \in I_{i,j} \\ 0, & if\ x \notin I_{i,j} \end{cases} \tag{11}$$

The number of membership function in the i^{th} dimension is:

$$m_i = \left\lfloor \frac{r_i}{2} \right\rfloor \tag{12}$$

where $\lfloor x \rfloor$ means the lower integer part of real x. The number of fuzzy rules:

$$R = \prod_{i=1}^{n} m_i \tag{13}$$

where n is the number of input dimensions. Each output can be defined also by four parameters: $v_{r,1}, v_{r,2}, v_{r,3}, v_{r,4}$ in the r^{th} rule. The task is to find the location of each λ and ν parameter. We suppose that the number of knots is given in each input dimension, so the number of rules is also given. The inference method must be discussed. The activation degree of the r^{th} rule (if the t-norm is the product):

$$w_r = \prod_{i=1}^{n} \mu_{i,r}(x_i) \tag{14}$$

w_r is the importance of the r^{th} rule if the input vector is $\mathbf{x}$ and $\mu_{i,r}(x_i)$ is the i^{th} membership function in the r^{th} rule. The r^{th} output is being cut in the height w_r. Then with the COG defuzzification method the output is calculated:.

$$y_{out} = \frac{\sum_{r=1}^{R} \int_{y \in \text{supp}\mu_r(y)} y\mu_r(y)dy}{\sum_{r=1}^{R} \int_{y \in \text{supp}\mu_r(y)} \mu_r(y)dy} \tag{15}$$

If this defuzzification method is used, the integrals can be easily computed. In (15) y_{out} will be the following:

$$y_{out} = \frac{1}{3} \frac{\sum_{r=1}^{R} (C_r + D_r + E_r)}{\sum_{r=1}^{R} 2w_r(v_{r,4} - v_{r,1}) + w_r^2(v_{r,3} + v_{r,1} - v_{r,4} - v_{r,2})}$$

$$C_r = 3w_r(v_{r,4}^2 - v_{r,1}^2)(1 - w_r)$$

$$D_r = 3w_r^2(v_{r,3}v_{r,4} - v_{r,1}v_{r,2}) \tag{16}$$

$$E_r = w_r^3(v_{r,3} - v_{r,4} + v_{r,1} - v_{r,2})(v_{r,3} - v_{r,4} - v_{r,1} + v_{r,2})$$

The algorithm needs to be modified, so that the ordering relation between the knots for each input dimension is maintained. In order to maintain the same search direction, the update vector, when violation of (7) is detected ($\lambda_{i+1}[k] < \lambda_i[k]$), is

reduced by a factor g so that the position of the $(i+1)^{th}$ knot is located half-way between the previous distance of the two corresponding knots:

$$g = \frac{\underline{\lambda}_{i+1}[k-1] - \underline{\lambda}_i[k-1]}{2(\underline{s}_{i+1}[k] - \underline{s}_i[k])} \tag{17}$$

Jacobean computation: (4) can be written as follows:

$$\underline{\underline{J}} = \begin{bmatrix} \frac{\partial y}{\partial \underline{\lambda}} & \frac{\partial y}{\partial \underline{v}} \end{bmatrix} \tag{18}$$

where

$$\frac{\partial y}{\partial \lambda} = \frac{\partial y}{\partial w_r} \frac{\partial w_r}{\partial \lambda_{r,i,k}} \tag{19}$$

where $\lambda_{r,i,k}$ is the k^{th} breakpoint of the i^{th} membership function in the r^{th} rule. So:

$$\underline{\underline{J}} = \begin{bmatrix} \frac{\partial y}{\partial w_r} \frac{\partial w_r}{\partial \lambda_{r,i,k}} & \frac{\partial y}{\partial v_{r,k}} \end{bmatrix} \tag{20}$$

From (14) can be seen that w_r depends on the membership functions , and each membership function depends only on four λ parameters. So, the derivatives of w_r will be:

$$\frac{\partial w_r}{\partial \lambda_{r,i^*,k}} = \frac{\partial w_r}{\partial \mu_{r,i}} \frac{\partial \mu_{r,i}}{\partial \lambda_{r,i^*,k}} = \prod_{\substack{i=1 \\ i \neq i^*}}^{n} \mu_{r,i} \frac{\partial \mu_{r,i^*}}{\partial \lambda_{r,i^*,k}} \tag{21}$$

The derivatives of the membership functions will be calculated as follows:

$$\begin{aligned}
\frac{\partial \mu_{r,i}(x_i)}{\partial \lambda_{r,i,1}} &= \frac{x_i - \lambda_{r,i,2}}{(\lambda_{r,i,2} - \lambda_{r,i,1})^2} N_{r,i,2}(x_i) \\
\frac{\partial \mu_{r,i}(x_i)}{\partial \lambda_{r,i,2}} &= \frac{\lambda_{r,i,1} - x_i}{(\lambda_{r,i,2} - \lambda_{r,i,1})^2} N_{r,i,2}(x_i) \\
\frac{\partial \mu_{r,i}(x_i)}{\partial \lambda_{r,i,3}} &= \frac{\lambda_{r,i,4} - x_i}{(\lambda_{r,i,4} - \lambda_{r,i,3})^2} N_{r,i,4}(x_i) \\
\frac{\partial \mu_{r,i}(x_i)}{\partial \lambda_{r,i,4}} &= \frac{x_i - \lambda_{r,i,3}}{(\lambda_{r,i,4} - \lambda_{r,i,3})^2} N_{r,i,4}(x_i)
\end{aligned} \tag{22}$$

$\frac{\partial y}{\partial w_r}$ and $\frac{\partial y}{\partial v_{r,k}}$ have to be also computed. From (16) the following can be written:

$$\frac{\partial y}{\partial par_r.} = \frac{1}{3} \frac{den \frac{\partial F_r.}{\partial par_r.} - num \frac{\partial G_r.}{\partial par_r.}}{(den)^2} \tag{23}$$

where $par_r = w_r, v_{r,1}, v_{r,2}, v_{r,3}, v_{r,4}$, *den* is the denominator and *num* is the numerator of (16), resp. $F_r.$ is the r* member of the sum in the numerator and $G_r.$ is the r* member in the denominator, $v_{r,k}$ is the kth breakpoint of the rth output membership function. The derivatives will be as follows:

$$\begin{aligned}
\frac{\partial F_r}{\partial w_r} &= 3(v_{r,4}^2 - v_{r,1}^2)(1-2w_r) + 6w_r(v_{r,3}v_{r,4} - v_{r,1}v_{r,2}) \\
&\quad + 3w_r^2\left[(v_{r,3} - v_{r,4})^2 - (v_{r,1} - v_{r,2})^2 \right] \\
\frac{\partial G_r}{\partial w_r} &= 2(v_{r,4} - v_{r,1}) + 2w_r(v_{r,3} + v_{r,1} - v_{r,4} - v_{r,2}) \\
\frac{\partial F_r}{\partial v_{r,1}} &= -6w_r v_{r,1} + 6w_r^2 v_{r,1} - 3w_r^2 v_{r,2} - 2w_r^3(v_{r,1} - v_{r,2}) \\
\frac{\partial G_r}{\partial v_{r,1}} &= -2w_r + w_r^2 \\
\frac{\partial F_r}{\partial v_{r,2}} &= -3w_r^2 v_{r,1} + 2w_r^3(v_{r,1} - v_{r,2}) \\
\frac{\partial G_r}{\partial v_{r,2}} &= -w_r^2 \\
\frac{\partial F_r}{\partial v_{r,3}} &= 3w_r^2 v_{r,4} - 2w_r^3(v_{r,4} - v_{r,3}) \\
\frac{\partial G_r}{\partial v_{r,3}} &= w_r^2 \\
\frac{\partial F_r}{\partial v_{r,4}} &= 6w_r v_{r,4} - 6w_r^2 v_{r,4} + 3w_r^2 v_{r,3} + 2w_r^3(v_{r,4} - v_{r,3}) \\
\frac{\partial G_r}{\partial v_{r,4}} &= 2w_r - w_r^2
\end{aligned} \tag{24}$$

The number of columns of $\underline{\underline{J}}$ will be $n_\lambda + n_\nu$. $n_\lambda = 4\sum_{i=1}^{n} m_i$ is the number of λ parameters and $n_\nu = 4R$ is the number of ν parameters.

4 The bacterial algorithm

Nature inspired some evolutionary optimisation algorithms suitable for global optimisation of even non-linear, high-dimensional, multimodal, and discontinuous problems. The original genetic algorithm (GA) was developed by Holland [2] and was based on the process of evolution of biological organisms. A more recent approach is the bacterial evolutionary algorithm (BEA). This gives an alternative to other algorithms, because it is simpler, and it is possible to reach lower error within a short time. This method includes not only the bacterial mutation operator - as in the pseudo-bacterial genetic algorithm (PBGA) – but also the gene transfer operation. These operations were inspired by the microbial evolution phenomenon. The bacterial mutation operation optimises the chromosome of one bacterium, the gene transfer operation allows to transfer information between the bacteria in the population.

4.1 The encoding method:

The membership functions are described by four parameters with the four breakpoints of the trapezium. Moreover the membership functions are identified by the two indices i and j. So, the membership function $A_{ij}(a_{ij},b_{ij},c_{ij},d_{ij})$ belongs to the ith rule and the jth input variable. $B_i(a_i,b_i,c_i,d_i)$ is the output membership function of the ith rule. The relative importance of the jth fuzzy variable in the ith rule:

$$A_{ij}(x_j) = \begin{cases} \dfrac{x_j - a_{ij}}{b_{ij} - a_{ij}}, & \text{if } a_{ij} < x_j < b_{ij} \\ 1, & \text{if } b_{ij} \le x_j < c_{ij} \\ \dfrac{d_{ij} - x_j}{d_{ij} - c_{ij}}, & \text{if } c_{ij} \le x_j < d_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

where $a_{ij} \le b_{ij} \le c_{ij} \le d_{ij}$ must hold.

So the encoding method of a fuzzy system with two inputs and one output, see in Fig. 2.

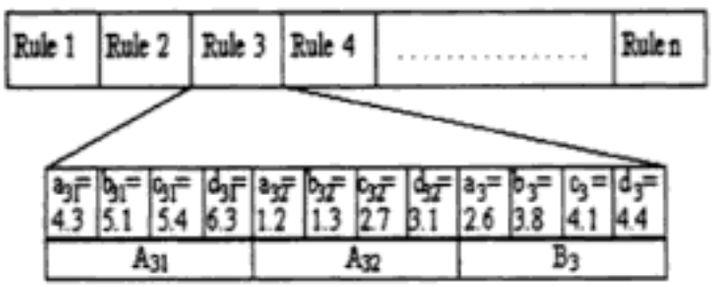


Fig. 2. Fuzzy rules encoded in a chromosome

For example, Rule 3 in Fig. 2 means:

$$\text{if } x_1 \text{ is } A_{31}(4.3,5.1,5.4,6.3) \text{ and } x_2 \text{ is } A_{32}(1.2,1.3,2.7,3.1) \text{ then } y \text{ is } B_3(2.6,3.8,4.1,4.4) \quad (26)$$

where A_{ij} and B_i mean the trapezoidal membership function with the four breakpoints.

4.2 The bacterial evolutionary algorithm:

4.2.1 Generating the initial population

First the initial (random) bacteria population is created. The population consists of n chromosomes (bacteria). This means that all membership functions in the chromosomes must be randomly initialised. The initial number of rules in one chromosome is N_{max}. So, $n(k+1)N_{max}$ membership functions are created, where k is the number of input variables in the given problem, and each membership function has four parameters.

4.2.2 Bacterial mutation

The bacterial mutation is applied to each chromosome one by one [5]. First, m -1 copies (clones) of the rule base are generated [4]. Then a certain part of the chromosome [5] is randomly selected and the parameters of this selected part are randomly changed in each clone (mutation). Next all the clones and the original bacterium are evaluated by the error criterion. The best individual transfers the mutated part into the other individuals. This cycle is repeated for the remaining parts, until all parts of the chromosome have been mutated and tested. At the end the best rule base is kept and the remaining m -1 are discharged. The above procedure is repeated once more, however, with different parameters [4]. It is an important question how long is one part suffering mutation and what is the degree of the mutation (expressed as the relative size in terms of the interval). This approach allows both selecting more than one membership function and fine-tuning. The number of mutated membership functions and the mutation degree are external parameters of the bacterial mutation. If selecting more than one membership

function is allowed then the local minima in the optimisation process can be avoided. Application of changing two or more membership functions at a time with fine-tuning in the first bacterial mutation and in the second step changing one membership function in the whole interval of the given variable was proposed in [4].

4.3 Gene transfer

The gene transfer operation allows the recombination of genetic information between two bacteria. (see Fig.3)

1. First the population must be divided into two halves. The better bacteria are called superior half, the other bacteria are called inferior half. [5]
2. One bacterium is randomly chosen from the superior half, this will be the source bacterium, and another is randomly chosen from the inferior half, this will be the destination bacterium.

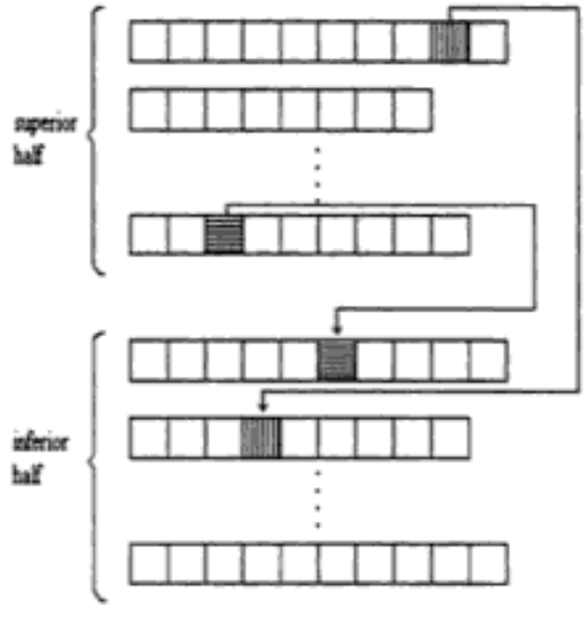


Fig .3. Gene transfer

3. A "good" part from the source bacterium is chosen and this part can overwrite a not-so-good part of the destination bacterium or simple be added. [5] A good part can be a fuzzy rule with a high degree of activation value. The activation value of a fuzzy rule can be calculated for example:

$$\overline{w}_i = \frac{1}{P}\sum_{j=1}^{P} w_i^{(j)} \tag{27}$$

where $\overline{w}_i$ is the mean activation value of the rule i^{th} rule, $w_i^{(j)}$ is the activation value of the i^{th} rule for the j^{th} pattern, P is the number of patterns. So the best

part of the source bacterium is the rule which has the greatest mean activation value.

4. 1, 2, and 3 are repeated for N_{inf} times, where N_{inf} is the number of "infections" per generation. [5].

4.4 Stop condition

If the population satisfies a stop condition or the maximum generation number is reached then the algorithm ends, otherwise it returns to the bacterial mutation step.

Between the bacterial mutation and the gene transfer step further operators can be applied, which optimise the rule number in the rule base. [6]

5 Clustering-Based Rule Extraction Technique

Recently, clustering-based approaches have been proposed for rule extraction [10, 11]. Most of the techniques use the idea of partitioning the input space into fixed regions to form the antecedents of the fuzzy rules. Although these techniques have the advantage of efficiency, they may lead to the creation of a dense rule-base that suffers from rule explosion. In general, the number of rules generated is t^d where d is the number of input dimensions and t is the terms per input. In this case, the number of rules generated grows exponentially with the increase of input dimensions. Due to this reason, the techniques are not suited for generating fuzzy rule-bases that have a large number of input dimensions.

Among the rule extraction techniques proposed in the literature, Sugeno and Yasukawa's [12] technique (abbreviated as SY method hereafter) is one of the earliest works that emphasize the generation of a sparse rule-base. The SY approach clusters only the output data and induces the rules by computing the projections to the input domains of the cylindrical extensions of the fuzzy clusters. This way, the method produces only the necessary number of rules for the input-output sample data (more details later). The paper [12] discusses the proposed technique at the methodological level leaving out some implementation details. The SY technique was further examined in [16,17] where additional readily implementable techniques are propose to complete the modeling methodology.

In this paper, a novel rule extraction technique that works on the projection of data and fuzzy clustering is introduced. The technique, known as projection-based fuzzy rule extraction (PB) , extends the idea of SY approach for fuzzy modeling. The goal of the rule extraction technique is to automate the construction of a fuzzy rule base from a set of input-output sample data. In the next paragraphs, a brief description of the original SY method is presented.

In the first step of SY modelling, the Regularity criterion [13] is used to assist in the identification of 'true' input variables that have significant influence on the output. The input variables that have less or no influence on the output are ignored

for the rest of the process. The true input variables are then used in the actual rule extraction process. The rule extraction process starts with the determinition of the partition of the output space. This is done by using fuzzy c-means clustering [14] (see section 6).

For each output fuzzy cluster B_i resulting from the fuzzy c-means clustering, a cluster in the input space A_i can be induced. The input cluster can be projected onto the various input dimensions to produce rules of the form:

$$\text{If x1 is } A_{i1} \text{ and x2 is } A_{i2} \text{ and } \ldots \text{ } x_n \text{ is } A_{in} \text{ then y is } B_i$$

However, it is remarked in the paper [12] that there can be more than one fuzzy cluster in the input space which corresponds to the same fuzzy cluster B_i. In this case more than one rule is formed with the same consequent. Suppose that two input clusters (A_i and A_j) are induced from the output cluster B_i, we obtain the following two rules:

$$\text{If x1 is } A_{i1} \text{ and x2 is } A_{i2} \text{ and } \ldots \text{ } x_n \text{ is } A_{in} \text{ then y is } B_i$$
$$\text{If x1 is } A_{j1} \text{ and x2 is } A_{j2} \text{ and } \ldots \text{ } x_n \text{ is } A_{jn} \text{ then y is } B_i$$

Unfortunately, no concrete procedures for determining the number of input clusters to be induced from an output cluster is discussed in the paper.

6 Fuzzy C-Means Clustering

Given a set of data, Fuzzy c-Means clustering (FCMC) performs clustering by iteratively searching for a set of fuzzy partitions and the associated cluster centers that represent the structure of the data as best as possible. The FCMC algorithm relies on the user to specify the number of clusters present in the set of data to be clustered. Given the number of cluster c, FCMC partitions the data $X = \{x_1, x_2, \ldots, x_n\}$ into c fuzzy partitions by minimizing the within group sum of squared error objective function as follows (eqn 28).

$$J_m(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} (U_{ik})^m \, \| x_k - v_i \|^2, \; 1 \le m \le \infty \tag{28}$$

where J_m(U,V) is the sum of squared error for the set of fuzzy clusters represented by the membership matrix U, and the associated set of cluster centers V. ||.|| is some inner product-induced norm. In the formula, $\|x_k - v_i\|^2$ represents the distance between the data x_k and the cluster center v_i. The squared error is used as a performance index that measures the weighted sum of distances between cluster centers and elements in the corresponding fuzzy clusters. The number m governs the influence of membership grades in the performance index. The partition becomes fuzzier with increasing m and it is proven that the FCMC algorithm

converges for any $m \in (1,\infty)$ [1]. The necessary conditions for eqn 28 to reach its minimum are

$$U_{ik} = \left(\sum_{j=1}^{c} \left(\frac{\| x_k - v_i \|}{\| x_k - v_j \|} \right)^{2/(m-1)} \right)^{-1} \qquad \forall i, \forall k \tag{29}$$

and

$$v_i = \frac{\sum_{k=1}^{n} (U_{ik})^m x_k}{\sum_{k=1}^{n} (U_{ik})^m} \tag{30}$$

In each iteration of the FCMC algorithm, matrix U is computed using eqn 29 and the associated cluster centers are computed as eqn 30. This is followed by computing the square error in eqn 28. The algorithm stops when either the error is below a certain tolerance value or its improvement over the previous iteration is below a certain threshold.

The FCMC algorithm cannot be used in situations where the number of clusters in a set of data is not known in advance. Since the introduction of FCMC, a reasonable amount of work has been done on finding the optimal number of cluster in a set of data. This is referred to as the cluster validity problem. The optimal number of clusters are determined by means of a criterion, known as the cluster validity index. Sugeno and Fukuyama proposes the following cluster validitiy index [15].

$$S(c) = \sum_{k=1}^{n} \sum_{i=1}^{c} (U_{ik})^m (\| x_k - v_i \|^2 - \| v_i - \bar{x} \|^2) \qquad 2 < c < n \tag{31}$$

where n is the number of data points to be clustered; c is the number of clusters; x_k is the k^{th} data, $\bar{x}$ is the average of data; v_i is the i^{th} cluster center; U_{ik} is the membership degree of the k^{th} data with respect to the i^{th} cluster and m is the fuzzy exponent. The number of clusters, c, is determined so that $S(c)$ reaches a local minimum as c increases. The terms $\| x_k - v_i \|$ and $\| vi - \bar{x} \|$ represent the variance in each cluster and variance between clusters respectively. Therefore, the optimal number of clusters is found by minimizing the distance between data to the corresponding cluster center and maximizes the distance between data in different clusters. Other cluster validity indexes can be found in [18]

7 Projection-Based Rule Extraction Technique

The main idea of the proposed PB rule extraction technique is similar to the SY approach [12]. Below, our novel technique will be discussed. The algorithm of the approach consists of 5 steps.

1. Perform fuzzy clustering on the output space. We suggest the use of Fuzzy-c Means Clustering with a suitable cluster validity index but remark that our algorithm does not place any restriction on the clustering technique used.
2. For each fuzzy output cluster B_i approximated, all the points belonging to the cluster are projected back to each of the input dimensions. For each dimension, fuzzy clustering is again applied to the projection of the points.
3. The previous step results in multiple 1D fuzzy clusters in each input dimension. For each fuzzy cluster, a trapezoidal cluster is approximated using the technique discussed in section 8.
4. Each of the n clusters $(C_{d1} - C_{dn})$ in the input dimension d, is a projection of the multi-dimensional input cluster to that input dimension. Next the clusters from individual dimensions are combined to form the multi-dimensional input cluster. The merging process is described in section 9.
5. For each of the multi-dimensional cluster identified, a rule can be formed. For example, if a multi-dimensional cluster is formed with $[C_{11}, C_{23}, C_{34},]$ for the points projected from output cluster B_i, we obtain the following rule:
   If x1 is C_{11} and x2 is C_{23} and x3 is C_{34} then y is B_i

   Where C_{dn} is the n^{th} cluster identified at input dimension d.

8 Trapezoidal Approximation Technique

A straightforward trapezoidal approximation technique from fuzzy clusters has been proposed in [16]. The trapeze approximation of the clustered raw data is proceeded in two steps. First, the convex hull of the original data set is determined, then the convex hull is approximated by a trapezoidal membership function. Figure 4 depicts the idea of the construction of trapezoidal membership function.

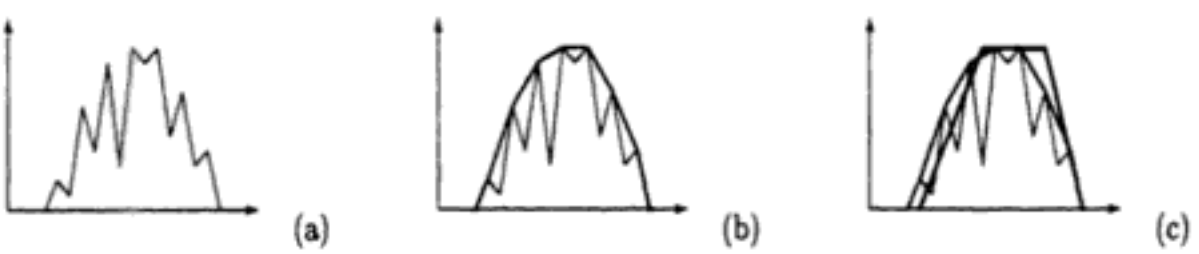


Fig.4. The positions of the membership functions

The algorithm is the following:

1. Determine μ_{min} and μ_{max} the minimum/maximum of the membership degree of the data point in the given cluster, x_{max} the frst point where the maximum μ_{max} is attained, further d_{min} and d_{max} the minimum/maximum of all the data values in the given cluster domain.
2. Set the boundaries of investigated interval to

$$m = \mu_{\min} \frac{r-1}{r} + \mu_{\max} / r \tag{32}$$

$$M = \mu_{\max} \frac{r-1}{r} + \mu_{\min} / r \tag{33}$$

   where r >1 is a given parameter, usually between 2 and 4.
3. Determine the parameters of the left slope, x_1 and x_2, as:
   a) Let us initialize x_j as the last data point which has smaller membership degree than m and xi be the next point of the convex hull:
      $\mu(x_j) < m;$ $\mu(x_{j+1}) = \mu(x_i) > m.$
      (If there is no such point x in the convex hull which satisfies $\mu(x) < m$ then x_j and x_i is the first two leftmost point of the convex hull). Further the parameters of the left $x_1 = d_{min}$ and $x_2 = d_{min}$.
   b) Let x^*_1 and x^*_2 be the location of the intersection made by the support and the core, respectively, with the line passing through the points $(x_j ; \mu(x_j))$ and $(x_i; \mu(x_i))$ (see Figure 5).
   c) If $x^*_1 > x_1$ then $x_1 := x^*_1$
   d) If $x^*_2 < x_2$ or $i = j + 1$ then $x_2 := x^*_2$
   e) If $x_{i+1} \leq x_{max}$ then let $i := i + 1$ and go to step (3b), otherwise continue
4. Determine the parameters of the left slope, x_3 and x_4, analogously as in the previous step.
5. Order the parameters according to $x_1 \leq x_2 \leq x_3 \leq x_4$.

For the convenience of later steps, we convert the trapezoidal clusters to ruspini partition as illustrated in figure 1.

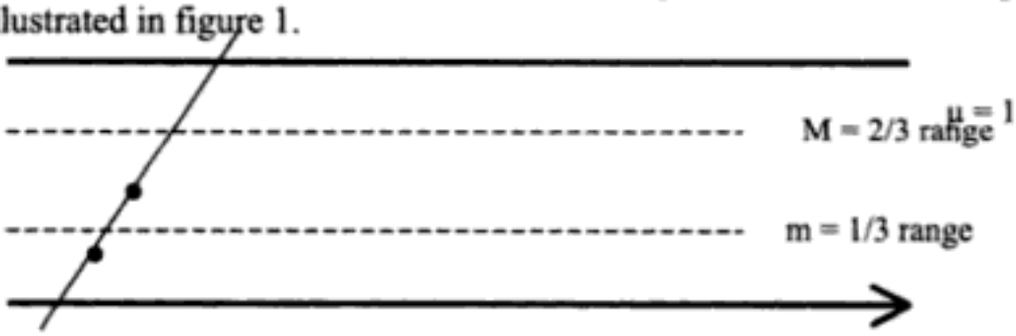


Fig.5. Determination of X^*_1 and X^*_2

9 Merging Scheme

The reconstruction of multi-dimensional clusters by combining the 1D clusters identified at each dimension can be problematic. Let t be the average number of clusters identified at each dimension. The total number of possible combination is t^d. Since the number of combination grows exponentially with the increase of dimensions, examining every combination of the 1D clusters is computationally infeasible. In this section, we propose a fast merging technique.

The merging process involves the use of a threshold t. The cluster in the multi-dimensional space is determined to be the region where the number of projected points in the region exceeds t. A point p is contained in the cluster C_i if $\mu_{Ci}(p) > \mu_{Cj}(p)$ for all $j \neq i$.

The 4-step algorithm is presented.

1. Find one of the multi-dimensional clusters C where the number of points that falls into all its projection exceeds the threshold t.
2. Remove all data points that are contained in the cluster C approximated.
3. Repeat steps 1 – 2 until no more cluster can be found.

The pseudo-C-code for step 1 of the algorithm is presented as follows.

10 PROCEDURE find_MD_cluster

```
Let U_i be the set of one-dimensional clusters in dimension i
Let mdCluster = [ ]
for i = 1 to k
   for each unit u ∈ U_i
      utemp = mdCluster x u
      if utemp is dense
        denseunit = utemp
        break
      end if
   end for
end for
```

For the convenience of discussion, we define [ ] as the zero-dimensional (empty) cluster where [ ] x C_i = C_i. The algorithm scans through each of the k dimensions to find one of the multi-dimensional clusters in the data giving the complexity O(k). Having identified a cluster, all the data points that is contained within the cluster is removed. The process is repeated until no more cluster can be found. The overall complexity is O(ck) where c is the total number of clusters in the data. Since the complexity of the algorithm is linear, it is computational feasible to deal with data with very large number of dimensions.

11 Conclusions

The Leveneberg-Marquardt and the bacterial evolutionary algorithm were described in this paper. The bacterial algorithm seems to be simpler and robust. In this method, the fuzzy systems can be described easier, and the algorithm allows using not only Ruspini-partition. Apart from the neural and evolutionary algorithm, a clustering based rule extraction technique has also been reported. The technique has the advatage of being computationally efficient.

References

1. L.A.Zadeh: Outline of a new approach to the analysis of complex systems and decision processes, IEEE Tr. Systems, Man and Cybernetics **3** (1973), pp. 28-44.
2. J.H.Holland: Adaptation in Nature and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, Cambridge, 1992.
3. L.J. Fogel, A.J.Owens, and M.J.Walsh: Artificial Intelligence through Simulated Evolution, Wiley, New York, 1966.
4. M.Salmeri, M.Re, E. Petrongari, and G.C.Cardarilli: A Novel Bacterial Algorithm to Extract the Rule Base from a Training Set, Dept. of Electronic Engineering, University of Rome, 1999.
5. N.E.Nawa, and T.Furuhashi: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, IEEE Tr. Fuzzy Systems **7** (1999), pp. 608-616.
6. J.Botzheim, B.Hámori, and L.T.Kóczy: Extracting trapezoidal membership functions of a fuzzy rule system by bacterial algorithm, 7th Fuzzy Days, Dortmund 2001,Springer-Verlag, pp. 218-227.
7. Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,* PhD. Dissertation, Appl. . Math., Harvard University, USA, 1974
8. Marquardt, D., An Algorithm for Least-Squares Estimation of Nonlinear Parameters, SIAM J. Appl. Math., 11, 1963, pp. 431-441
9. A.E.Ruano, C. Cabrita, J.V.Oliveira, L.T.Kóczy, D. Tikk: Supervised Training Algorithms for B-Spline Neural Networks and Fuzzy Systems, Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, 2001.
10. Wong, K.W., Fung, C.C., and Wong, P.M. A self-generating fuzzy rules inference systems for petrophysical properties prediction. in Proceedings of IEEE International Conference on Intelligent Processing Systems. 1997. Beijing.
11. Wang, L.X. and Mendel, J.M., Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man, and Cybernetics, 1992. **22**(6): p. 1414-1427.

12. Sugeno, M. and Yasukawa, T., A fuzzy-logic-based approach to qualitative modeling. IEEE Transactions on Fuzzy Systems, 1993. **1**(1): p. 7-31.
13. Ihara, J., Group method of data handling towards a modelling of complex systms - IV. Systems and Control (in Japanese), 1980. **24**: p. 158-168.
14. Bezdek, J.C., Pattern Reconition with Fuzzy Objective Function Algorithms. 1981, New York: Plenum Press.
15. Fukuyama, Y. and Sugeno, M. A new method of choosing the number of clusters for fuzzy c-means method. in Proceedings of the 5th Fuzzy System Symposium. 1989.
16. Tikk, D., Gedeon, T. D., Koczy, L. T., and Biro, G. Implementation details of problems in Sugeno and Yasukawa's qualitative modelling. Research Working Paper RWP-IT-02-2001, School of Information Technology, Murdoch University, Perth, W.A., 2001. P. 17.
17. Wong, K.W., Kóczy, L.T., Gedeon, T.D., Chong, A., Tikk, D. (2001) "Improvement of the Clusters Searching Algorithm in Sugeno and Yasukawa's Qualitative Modeling Approach" in Reusch, B. (Ed), Computational Intelligence: Theory and Applications, Springer-Verlag, Berlin, Proceedings of 7th Fuzzy Days in Dortmund – International Conference on Computational Intelligence, October 2001, Dortmund, pp. 536-549.
18. Yang, M.S., Wu, K.L., A New Validity Index For Fuzzy Clustering, in Proceedings of IEEE International Conference on Fuzzy Systems, December, Melbourne, 4 pages.