Mathware & Soft Computing 7 (2000) 159-173

A Defuzzification Based New Algorithm for the Design of Mamdani-Type Fuzzy Controllers

J.J. Saade
ECE Department, FEA, American University of Beirut,
P. O. Box: 11-0236, Beirut, Lebanon
jsaade@aub.edu.lb

Abstract

This paper presents a new learning algorithm for the design of Mamdani-type or fully-linguistic fuzzy controllers based on available input-output data. It relies on the use of a previously introduced parametrized defuzzification strategy. The learning scheme is supported by an investigated property of the defuzzification method. In addition, the algorithm is tested by considering a typical non-linear function that has been adopted in a number of published research articles. The test stresses on data-fitting, function shape representation, noise insensitivity and generalization capability. The results are compared with those obtained using neuro-fuzzy and other fuzzy system design approaches.

1 Introduction

Fuzzy controllers design based on interviewing the human expert and transforming his knowledge into linguistic terms and fuzzy inference rules has often led to tedious and time-consuming trial and error design procedures [1,2]. This has been mainly due to the fact that the expert is usually unable to describe linguistically the kind of actions he takes in a particular situation [3,4]. As a result, most of the recent research in the above-mentioned design or modeling area has been centered on devising automatic techniques to build fuzzy controllers using a set of numerical input-output data representing the expert's control actions [3]. The majority of these data-driven techniques rely on the use of Takagi-Sugeno type fuzzy-controllers [3] and combined fuzzy-neural-network [5-9], fuzzy-clustering, fuzzy-partition and genetic algorithm approaches [2,10-16]. Takagi-Sugeno type controllers, however, are not fully linguistic and the use of neural-network and other learning algorithms has often led to final systems that are difficult to interpret linguistically [2,8,10,13].

In a previous study, a new defuzzification strategy has been developed [17]. Particularly important in this strategy is its containment of a free parameter, which can be used for adaptation and modification of the crisp defuzzified values to help

fit the problem of concern. Such parametrization is among the features stated in [18,19] of a desired defuzzification method. In this study, a property of the noted defuzzification method is investigated. It is also used to support the adaptation aspects of the method (Section 2), and introduce a new learning algorithm for the design or construction of fully linguistic fuzzy controllers based on available input-output data (Section 3). The algorithm requires the setting of an initial fuzzy controller. Then a consistent modification of the defuzzification parameter and rules consequents is to be performed in order to reduce the value of some error function, resulting from the approximation of the available data, and obtain a final fuzzy system. In Section 4, the algorithm is tested by considering a typical non-linear function that was adopted in published research. The test stresses on data-fitting, function shape representation, noise insensitivity and generalization capability. The results are also compared with those obtained using fuzzy-neural, fuzzy clustering and partition techniques. Concluding remarks are offered in Section 5.

2 The new defuzzification strategy and its property

Consider a two-input, one-output fuzzy inference system formed by a set of N if-then fuzzy inference rules. Let the j-th rule, $1 \leq j \leq N$, be represented by:

$$\textit{If } x_1 \textit{ is } A_j \textit{ and } x_2 \textit{ is } B_j, \textit{ then } z \textit{ is } C_j \,.$$

In the above rule, x_1 and x_2 are the input variables and z is the output variable of the fuzzy system. Also, A_j and B_j are fuzzy sets defined over x_1 and x_2 respectively. C_j is a fuzzy set defined over z. The "if " part of the rule is called rule "antecedent" and the "then" part is the rule "consequent." The fuzzy output, denoted by $C_{0i}(z)$ and corresponding to some crisp input pair (x_{1i}, x_{2i}), can be obtained using the compositional rule of inference (CRI) [20] as follows:

$$C_{0i}(z) = \max_{1 \leq j \leq N} [A_j(x_{1i}) \wedge B_j(x_{2i}) \wedge C_j(z)]. \qquad (1)$$

The fuzzy OR, AND and THEN are respectively represented by maximum, minimum and minimum operations. Other operations, such as sum and product, can also be used [9]. Also, Equation (1) can be generalized easily to systems with more than two input variables.

Now, the new defuzzification strategy, which was established by generalizing the standard decision-making criteria for ranking intervals to fuzzy sets [17,21], applies to the normalized version of $C_{0i}(z)$, denoted $C_{0in}(z)$, as follows:

$$F_\delta [C_{0in}(z)] = \int_0^1 [\delta\, c_1(\alpha) + (1-\delta)\, c_2(\alpha)]\, d\alpha \,. \qquad (2)$$

The normalization of a subnormal fuzzy set can be obtained by dividing the fuzzy set membership function by its highest membership grade [22]. In Equation (2), δ is a parameter that takes values in the interval [0,1]. $c_1(\alpha)$ and $c_2(\alpha)$ are the lower and upper values of the bounded interval representing the α-level set of $C_{0in}(z)$ when it is convex and has a bounded support (see Property 1 and its proof). Whether $C_{0in}(z)$ is still with a bounded support but non-convex, then we consider

$c_1(\alpha)$ and $c_2(\alpha)$ as the lower and upper values of the α-level set of $C_{0in}(z)$. Thus, the gap in $(C_{0in})_\alpha$, when it occurs for any α-value, is to be bridged to form a bounded interval. This consists of filling the cavity(ies) that exists in the membership function of $C_{0in}(z)$ to convert the non-convex fuzzy set into a convex one. This procedure is in fact consistent with the basic definitions of left and right sets used in [21] to reformulate the classical criteria for ranking intervals and make them applicable to fuzzy sets. This was done by replacing the interval characteristic function by the fuzzy set membership function.

It is to be noted here that the development of the strategy in Equation (2) was done in [17] by considering a relative perspective for defuzzification. This was mainly motivated by the interdependence between the fuzzy outputs and the need to defuzzify them with respect to each other while permitting flexibility to help meet various performance objectives. The noted strategy turned out to possess features stated in [18,19] of a desired defuzzification method, particularly the containment of a free parameter, and have a useful property, which is addressed in this section. This property is also shown to support the adaptation aspects of the strategy represented by its ability to introduce changes in the location of the controller input-output characteristic (control surface) relative to the range of the output variable, and in the steepness and shape of this surface. Furthermore, the Property is used to serve the main objective of this study. It is the production of a learning procedure where the concern is the consistent modification of the defuzzification parameter and rules consequents to reduce the data approximation error and deliver a final fuzzy system (Section 3).

Property 1 Given any convex and normal fuzzy set C defined over the set of real numbers, ℜ, (C is not a crisp number) and such that C has a bounded support, then $F_\delta(C)$ is a linear and strictly decreasing function of δ. Also, $F_\delta(C)$ decreases from $F_0(C)$ to $F_1(C)$ with [$F_0(C)$ - $F_1(C)$]=(Area under C).

Proof

$F_\delta(C)$ as in Equation (2) can be written as:

$$F_\delta(C) = -\delta\int_0^1[c_2(\alpha)-c_1(\alpha)]d\alpha + \int_0^1 c_2(\alpha)d\alpha$$
$$= -[F_0(C)-F_1(C)]\,\delta + F_0(C), \qquad (3)$$

where

$$F_0(C) = \int_0^1 c_2(\alpha)d\alpha \qquad (4)$$

and

$$F_1(C) = \int_0^1 c_1(\alpha)d\alpha \qquad (5)$$

Now, since the fuzzy set C is convex, then the α-level set of C, denoted C_α, is convex for any $\alpha \in (0,1]$ [23]. Further, since C is defined over ℜ, then C_α is an interval. But C_α is a subset of the support of C for any $\alpha \in (0,1]$ [24]. Hence, taking C_α as the closure of the support of C for α=0 makes C_α a bounded interval for any $\alpha \in [0,1]$. As a result, for each $\alpha \in [0,1]$, $c_1(\alpha)$ and $c_2(\alpha)$, which are the lower and

upper values of C_α ($c_1(\alpha) < c_2(\alpha)$), are unique and bounded. They are, therefore, single-valued bounded functions of α. Consequently, $F_0(C)$ and $F_1(C)$ exist and are such that $F_1(C) < F_0(C)$. By Equation (3), $F_\delta(C)$, for any $\delta \in [0,1]$, also exists and it is a linear and strictly decreasing function of δ. The highest value of $F_\delta(C)$ over its domain, i.e., the interval [0,1], is attained when δ=0 and the lowest value is attained when δ=1. Further, $F_0(C)$ and $F_1(C)$ as in Equations (4) and (5) are respectively the areas between the decreasing and increasing parts of the membership function of C and the membership axis passing by zero. Hence [$F_0(C)$ - $F_1(C)$]=(Area under C) and Equation (3) can be written as

$$F_\delta(C) = -(\textit{Area under } C)\,\delta + F_0(C). \qquad (6)$$

We note here that when C is a crisp number, denoted by p, say, then $c_1(\alpha) = c_2(\alpha) = p$ for any $\alpha \in (0,1]$. Hence, $F_1(C) = F_0(C) = p$ and $F_\delta(C) = F_\delta(p) = p$. ∈

Remark

In a fuzzy inference system, the input-output characteristic is obtained from the defuzzified values of output fuzzy sets resulting from different crisp input points distributed in the input space. Since the fuzzy outputs are obtained using Equation (1), it becomes easy to see that these fuzzy outputs and hence the areas under their membership functions are generally different. Consequently, by Equation (6), the pair-wise separation between the defuzzified values of these fuzzy outputs will generally change when δ varies. Hence, Property 1 validates what was previously stated in this section regarding the effects of parameter change on the input-output characteristic of a fuzzy inference system.

The following example also demonstrates the adaptation aspects of the strategy in Equation (2). Let a two-input, one-output fuzzy inference system with rules as shown below be considered:

If x_1 *is* A_1 *and* x_2 *is* B_1, *then z is* C_1
If x_1 *is* A_1 *and* x_2 *is* B_2, *then z is* C_1
If x_1 *is* A_1 *and* x_2 *is* B_3, *then z is* C_2
If x_1 *is* A_2 *and* x_2 *is* B_1, *then z is* C_1
If x_1 *is* A_2 *and* x_2 *is* B_2, *then z is* C_2
If x_1 *is* A_2 *and* x_2 *is* B_3, *then z is* C_3
If x_1 *is* A_3 *and* x_2 *is* B_1, *then z is* C_2
If x_1 *is* A_3 *and* x_2 *is* B_2, *then z is* C_3
If x_1 *is* A_3 *and* x_2 *is* B_3, *then z is* C_3 (7)

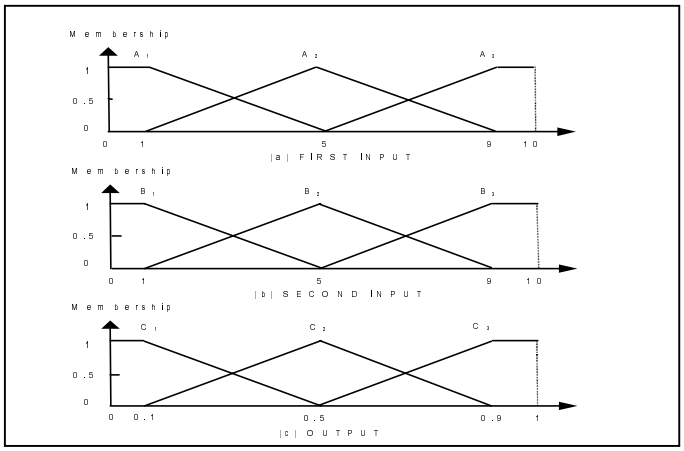
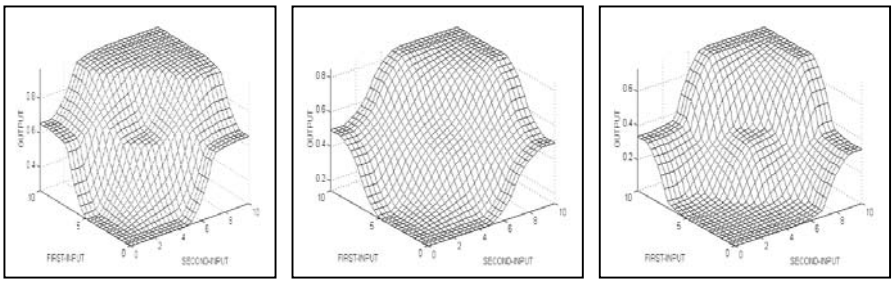


Figure 1. Input and output membership functions of the fuzzy system given in Eq. (7).

The membership functions of the input and output fuzzy sets assigned over the input and output variables of this fuzzy system are shown in Figure 1. The application of Equation (2), with δ=0.1, 0.5 and 0.9, to the fuzzy system in Equation (7) results in the control surfaces shown in Figure 2. What has been previously noted regarding the change in the location and shape of the input-output surface can be observed in this figure.



(a) (b) (c)

Figure 2. Input-output characteristics of the fuzzy system given in Equation (7) for different values of δ. The Cases are for δ=0.1, δ=0.5 and δ=0.9.

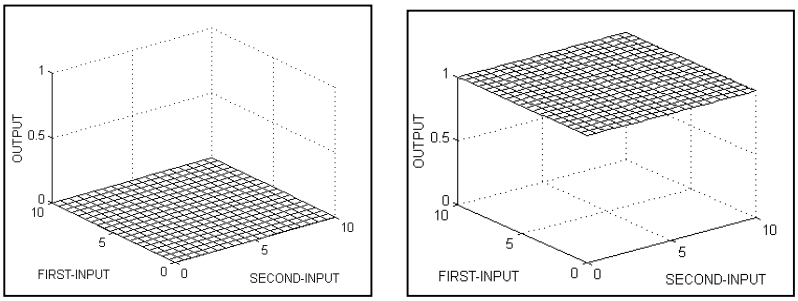
3 The design algorithm and its requirements

In this section, the algorithm that can be used to design fuzzy controllers using a given set of input-output data is introduced. It is based on the property of the defuzzification function given in Section 2. Emphasis is also placed on the requirements needed to construct the initial fuzzy system. As will be seen, Property 1 permits the conception of a learning scheme whereby a consistent modification of the defuzzification parameter and rules consequents is implemented in order to arrive at a final fuzzy system. This is done through the reduction of the data approximation error.

It is assumed that the system designer is able to specify the input and output variables of the fuzzy controller and the ranges of these variables. Having this accomplished, then overlapping membership functions for the input and output fuzzy sets are assigned to cover the ranges of the variables of concern. In terms of overlap, it is recommended, as in most fuzzy applications, that the membership functions, assigned over a single variable, be such that the membership grades of any crisp value sum to 1 (Figure 1). This will also be shown helpful in the described learning process. Further, the number of membership functions to start with, especially input ones, is to be as small as possible (this reduces the size of the rule-base as can be seen below) and their shape as simple as possible. Whether the error value and surface that result after learning are not as desired then an increase in the number and/or change in the shape of the input and output fuzzy sets could be considered (Section 4). The number of output fuzzy sets should be increased first to keep a small rule-base size if possible. Also, the number of these sets has a direct

effect on the data approximation error as will be seen in the described learning procedure.

Once the input membership functions are assigned, then all combinations of input fuzzy sets are considered to form the antecedent parts of the rules. Due to the learning process described below, the initial rules consequents need to be equal to the left-most output fuzzy set. This set is required to be formed by a decreasing part or a flat and decreasing part (Figure 1). This allows the defuzzified value of any fuzzy output obtained by Equation (1), and through the application of Equation (2) with δ=1, to be equal to the smallest value of the output range (Equation (5)). For the same reason, the right-most output fuzzy set is to be formed by an increasing part, or a flat and an increasing part (see also Figure 1). This allows the defuzzified value of any fuzzy output obtained using the right-most set to reach the highest value of the range of the output variable when δ =0 (Equation (4)). Figures 3(a) and 3(b) show the input-output surfaces of the fuzzy system in Equation (7) when all rules consequents are respectively set to C_1 (δ=1) and then to C_3 with δ=0.



(a) (b)

Figure 3. Input-output surfaces of the fuzzy system given in Eq. (7). Case (a) rules consequents are set to C_1 and δ=1. Case (b) consequents are set to C_3 and δ=0.

Given a set of input-output data pairs in the form ($\underline{x}_i$, z_{id}), with i=1, 2, 3,…,n and $\underline{x}_i = (x_{1i}, x_{2i}, x_{3i}, ..., x_{pi})$, where p is the number of input variables, the learning process starts with an initial fuzzy system constructed in the above-noted manner. The algorithm, whose flow-chart is shown in Figure 4, computes the fuzzy outputs C_{0i} for all $\underline{x}_i$, i=1, 2, 3, …, n using the CRI (Equation (1)) and then defuzzifies their normalized and convex versions using Equation (2) when δ =1. Here, all the defuzzified values are equal to the smallest value of the output range. Hence, given that z_{id} are all greater than the smallest value of the output range (this should always be the case), then $F_1[C_{0in}(z)] < z_{id}$, for all i=1, 2, 3, …, n. For these defuzzified values, the error E is computed using some error function; such as the mean-square error (MSE) [10], root mean-square error (RMSE) [12], etc., and compared with a desired error value, denoted E_d . If $E \le E_d$, then the learning stops. Otherwise, δ is decreased from 1 to 0 by passing through discrete intermediate values. For each δ , the error is computed and compared with E_d . Note here that by Property 1 the decrease in δ results in an increase in the defuzzified values of the fuzzy outputs. These values are then made closer to the desired outputs and this causes a reduction in the value of the error. Whether the

change in δ has led to the satisfaction of the error goal, that is, $E \le E_d$ has been achieved for some $\delta \in [0,1]$, then the learning stops. Otherwise, the algorithm starts again from $\delta = 1$ but with a new set of rules.

The new set of rules is obtained by raising each rule consequent by one fuzzy set (the reason for this raise is explained below). This, however, might lead to a violation of the inequality $F_1[C_{0in}(z)] < z_{id}$. Whether the noted inequality is violated for all i=1,2,3,…,n, then, again by Property 1, the decrease in δ will not serve the error reduction objective. Whether the violation is for some of the data points, then the decrease in δ would not serve the error reduction in the same manner as when the inequality is satisfied for all i. Thus, if the inequality is violated, it needs to be reestablished. This can be done by repeatedly lowering the consequents of the rules, which trigger one fuzzy output whose defuzzified value for $\delta = 1$ is greater than its desired counterpart. Once all defuzzified values become again smaller than the desired ones, and the obtained new rules differ from the initial rules, then δ will be decreased from 1 to 0 and for each δ the error is computed and compared with E_d.

The learning process described above is to be repeated until either the error goal is satisfied or no more raise in the rules consequents is possible (when all rules consequents reach the highest (right-most) output fuzzy set). The learning needs also to stop when the above-noted raise and lowering of the rules consequents result in rules that have already been obtained. When the learning stops, the algorithm delivers the final fuzzy system with the least error value that can be obtained under the described procedure, the error and the final δ value.

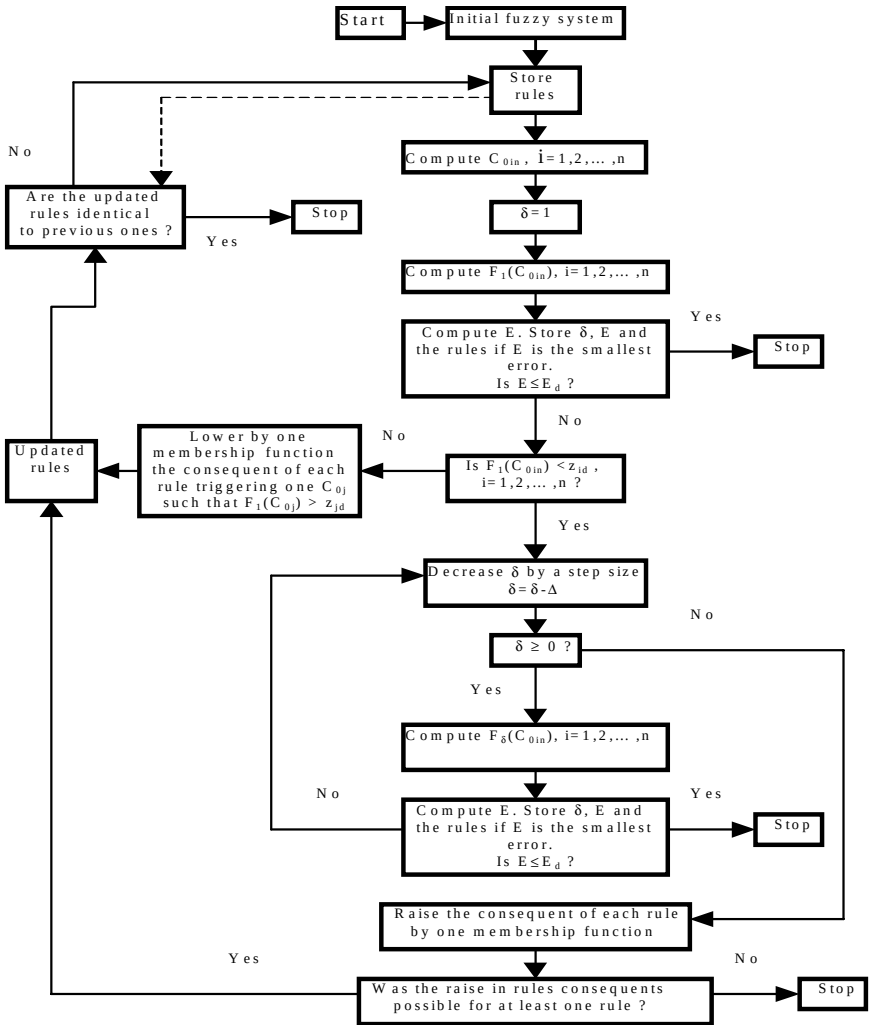


Figure 4. Flow-chart of the design algorithm described in Section 3.

What follows is an explanation of the reasons behind the above-mentioned raise in the rules consequents. Of course, this raise cannot be explained independently of the other steps in the learning procedure, particularly, the subsequent lowering of some of these consequents to reestablish the inequality $F_1[C_{0in}(z)] < z_{id}$ and obtain new rules to which the δ decrease is applied. When all the rules consequents are equal to the left-most output fuzzy set, all fuzzy outputs resulting from the data input tuples through the application of Equation (1) are truncated versions of this left-most set. Thus, upon normalization and application of Equation (2), the δ decrease leads to defuzzified values which can reach no more than the value that is just below the upper limit of the support of the left-most output fuzzy set (see Equation (4)). Hence, in order to allow the defuzzified values to reach more desired

outputs, which are normally distributed over most of the output range (this can be made to happen by consistency between the available desired outputs and the specified output range), a raise in the rules consequents is applied.

After the first raise (the one applied to the initial rules consequents), the new set of rules will be formed by consequents equal to the second left-most output fuzzy set (if no lowering is necessary) or by consequents some of which are the second and others are the first left-most output fuzzy set. This makes obtaining fuzzy outputs formed by truncating the second or the first and second left-most sets possible. This possibility becomes even higher when the input points are distributed over most of the input space (this can also be made to happen by consistency between the crisp input points and the specified range of the corresponding input variable) and they are large in number. Since, in such a case and with the previously-noted overlap of the input membership functions and assigned rules antecedents, most or even all the rules will be triggered by the input tuples. Under such circumstances, the application of Equation (2) to the normalized outputs leads to defuzzified values which can reach, through the δ decrease, the value that is just below the upper limit of the support of the second left-most output fuzzy set. Given the previously-mentioned overlap of the output fuzzy sets, this upper limit is higher than that of the left-most set. Thus, more desired outputs can be reached by defuzzification and δ decrease. The repetition of the described procedure of raise and lowering of the rules consequents is supposed to lead to having additional desired outputs with each being within the range of variation of its corresponding defuzzified value. Thus, an enhancement of the chances for error reduction and satisfaction of the error goal results.

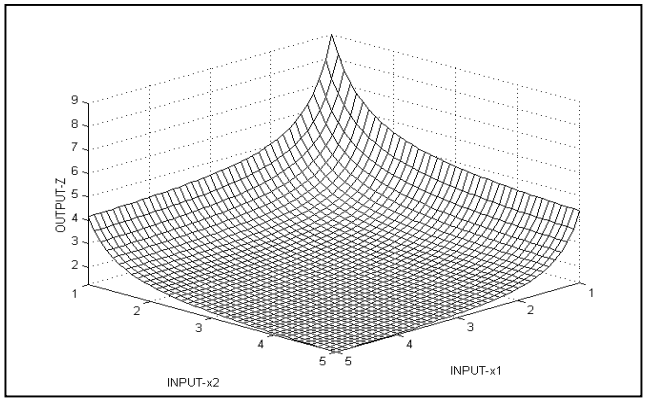


Figure 5. Input-output surface of the non-linear function given in Equation (8).

4 Example

In this example, the following two-variable non-linear function is considered:

$$z = f(x_1, x_2) = (1 + x_1^{-2} + x_2^{-1.5})^2, 1 \le x_1, x_2 \le 5. \qquad (8)$$

The plot of the input-output surface of this function is shown in Figure 5. This function was first considered by Sugeno and Yasukawa in [10] and then in [12] and

[13]. 50 input-output data points (listed in [10]) were used in all the noted references.

In [10], a 6-rule fuzzy system was determined based on fuzzy clustering with 0.318 as a MSE value. The error was then reduced to 0.01 using position gradient. In [12], which is also a study that relies on fuzzy clustering and aimed at the rapid prototyping of fuzzy models based on data, the best-obtained MSE was 0.231. 5 rules were considered. The use of fuzzy partition in [13] gave a 6-rule fuzzy system with 0.351 as a MSE value. This was then reduced to 0.005 by initiating and training a fuzzy neural network.

In terms of the algorithm introduced in this study, the same 50 data points were used. As in Figures 6(a) and 6(b), 3 membership functions were considered over each of the input variables. Hence, 9 inference rules representing all the combinations of the input fuzzy sets were adopted. The number of output fuzzy sets, which was first attempted, was 3. However, compared to the above-noted error figures, the obtained MSE was higher. The error value of 0.216 was obtained with 4 output fuzzy sets whose membership functions are shown in Figure 6(c). The final obtained fuzzy rules are as listed below and the final δ value is 1. The input-output surface is shown in Figure 7.

If x_1 *is* A_1 *and* x_2 *is* B_1, *then z is* C_3 *If* x_1 *is* A_1 *and* x_2 *is* B_2, *then z is* C_3

If x_1 *is* A_1 *and* x_2 *is* B_3, *then z is* C_3 *If* x_1 *is* A_2 *and* x_2 *is* B_1, *then z is* C_3

If x_1 *is* A_2 *and* x_2 *is* B_2, *then z is* C_2 *If* x_1 *is* A_2 *and* x_2 *is* B_3, *then z is* C_2

If x_1 *is* A_3 *and* x_2 *is* B_1, *then z is* C_3 *If* x_1 *is* A_3 *and* x_2 *is* B_2, *then z is* C_2

If x_1 *is* A_3 *and* x_2 *is* B_3, *then z is* C_2 (9)

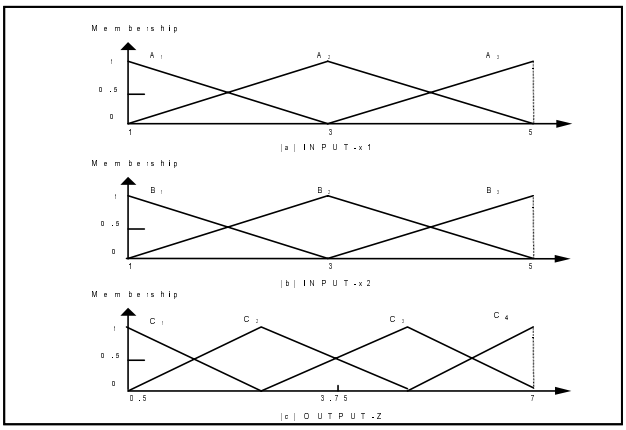


Figure 6. Membership functions assigned over the input and output variables of the non-linear function given in Equation (8).

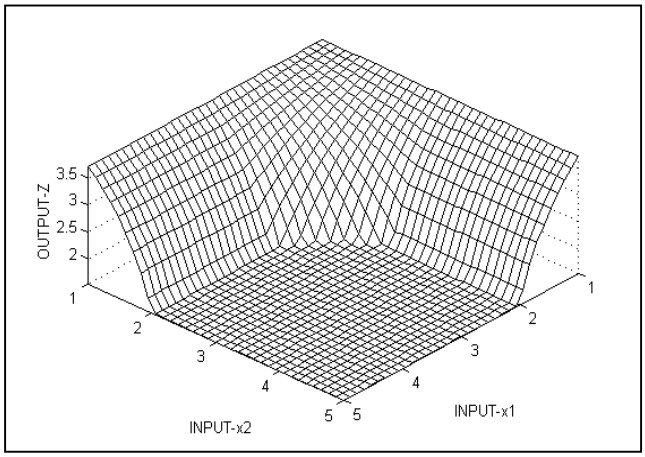


Figure 7. Input-output surface of the fuzzy system in Equation (9)

Hence, the presented results show that the proposed algorithm can provide a better data-fitting than the clustering and fuzzy partition methods [10,12,13]. The data-fitting, however, is inferior to that of the combined fuzzy-partition-neural-network [13] and clustering-gradient-position [10].

Knowing that a smaller error value obtained in the approximation of data does not necessarily imply a better function shape representation, and since the surfaces of the obtained fuzzy systems in [10,12,13] were not given, we choose to compare the results of this example with those obtained by ANFIS [8]. It is a powerful data-driven fuzzy-system construction methodology based on a combined gradient-descent and least-squares and available under a MATLAB tool-box. Also the presented approach will be tested for noise insensitivity and generalization capability. This is taken in the sense of extrapolation to regions where there are missing data. ANFIS is not structured to account for various configurations related to this type of generalization.

The use of the same 50 data pairs [10] and 3 triangular membership functions over each of the 2 input variables in ANFIS gave a nine-rule fuzzy system with 0.0303 as an MSE under 100 epochs and 0.00001 initial step-size. The input-output surface is shown in Figure 8. These are the best results obtained after attempting various combinations of epoch number and step-size value. Since the error value is smaller than 0.216 obtained in our approach, then ANFIS provided a better data-fitting. The comparison of Figures 7 and 8, however, reveals that the proposed approach has a better representation of the shape of the non-linear function.

Concerning the noise insensitivity issue, we considered 3 stages of modification of output values in the 50 points listed in [10]. In each stage, 4 output values were modified to result in 4 noisy input-output pairs, denoted as a set, not satisfying the function in Equation (8) (see Table 1). First, set 1 was used in addition to the remaining 46 noise-free data pairs. ANFIS gave a nine-rule fuzzy system with 0.0422 MSE and input-output surface shown in Figure 9(a). In stage 2, sets 1 and 2 were used in addition to the remaining 42 noise-free data pairs. A 9-rule fuzzy system was obtained by ANFIS

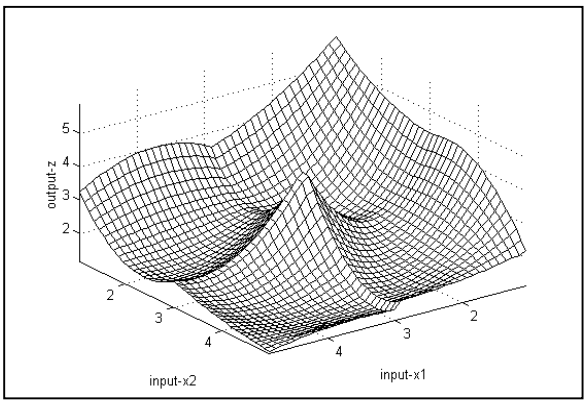


Figure 8. Input-output surface of the fuzzy system obtained from ANFIS using the 50 data pairs.

with 0.0415 MSE and surface given in Figure 9(b). In stage 3, sets 1, 2 and 3 were used in addition to the remaining 38 noiseless data. The obtained 9-rule fuzzy system had 0.1037 MSE and surface shown in Figure 9(c).

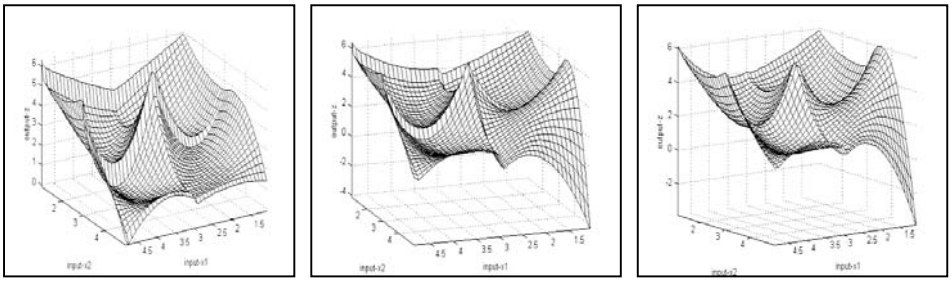
All the above-noted 3 cases of noisy and noise-free data pairs were entered into the algorithm proposed in this study. The resulting 9-rule fuzzy system was always as in Equation (9) and with δ=1. The resulting error values were respectively 0.3842, 0.4583 and 0.5056. The input-output surface is just the one shown in Figure 7. The comparison of Figure 7 with Figures 9(a), (b) and (c) reveals that the presented approach has a better noise insensitivity than ANFIS. This result is also supported by the MSE values at the 4, 8 and 12 noisy points. They are respectively 0.0352, 0.0185, 0.1762 for ANFIS and 2.1960, 1.5724, 1.2606 for the given algorithm.

Regarding the error values obtained by considering the original 50 noise-free data [10], ANFIS gave 0.2292, 0.2925 and 0.3217 respectively after introducing the 4, 8 and 12 noisy data pairs indicated in Table 1. Comparison of these error values with 0.216 (obtained using the presented methodology), and also Figure 7 with Figures 9(a), (b) and (c), shows that ANFIS has in this example an inferior performance efficiency when the learning is based on noisy data. This efficiency is measured by considering the noise-free data error, noise insensitivity and function shape representation.

In terms of testing the generalization capability of the presented approach, sets of data points from among those listed in [10] and located in specific input space regions were excluded in succession and the remaining data pairs were entered into the presented algorithm. This was done in order to see whether the algorithm returns the same fuzzy system and thus the same input-output characteristic obtained with the whole set of data when a part of the data is missing. First, data pairs such that $1< x_1 <2.5$ and $3.5< x_2 <5$ were eliminated. This resulted in the exclusion of 5 data points. The remaining 45 points

#	x1	x2	z		#	x1	x2	z		#	x1	x2	z		#	x1	x2	z	
1	1.4	1.8	3.7		14	1.67	2.81	2.47		27	2.71	4.13	1.6		39	4.47	3.66	1.42	
2	4.28	4.96	1.31		15	2.03	1.88	3.7	***	28	1.78	1.11	4.7		40	1.35	1.76	3.91	
3	1.18	4.29	3.35		16	3.62	1.95	2.08		29	3.61	2.27	1.9		41	1.24	1.41	5.05	
4	1.96	1.9	2.7		17	1.67	2.23	2.75		30	2.24	3.74	2.8	**	42	2.81	1.35	1.97	
5	1.85	1.43	3.52		18	3.38	3.7	1.51		31	1.81	3.18	3	**	43	1.92	4.25	3	**
6	3.66	1.6	2.46		19	2.83	1.77	3.7	***	32	4.85	4.66	1.3		44	4.61	2.68	3.6	*
7	3.64	2.14	1.95		20	1.48	4.44	2.44		33	3.41	3.88	1.5		45	3.04	4.97	1.44	
8	4.51	1.52	4	*	21	3.37	2.13	1.99		34	1.38	2.55	3.1		46	4.82	3.8	3.1	*
9	3.77	1.45	2.7		22	2.84	1.24	4.1	***	35	2.46	2.12	2.2		47	2.58	1.97	2.29	
10	4.84	4.32	1.33		23	1.19	1.53	4.99		36	2.66	4.42	1.6		48	4.14	4.76	1.33	
11	1.05	2.55	4.63		24	4.1	1.71	2.27		37	4.44	4.71	1.3		49	4.35	3.9	1.4	
12	4.51	1.37	4.2	*	25	1.65	1.38	3.94		38	3.11	1.06	4.1		50	2.22	1.35	4	***
13	1.84	4.43	2.7	**	26	2	2.06	2.52		* denotes set 1, ** denotes set 2 & *** denotes set 3.									

Table 1. 50 noisy and noise-free data used in the Example



(a) (b) (c)

Figure 9. Input-output surfaces of the fuzzy systems obtained from ANFIS.

were used for training in the algorithm. The final fuzzy system turned out to be as in Equation (9) with the same δ value and MSE=0.2355. Second, data points such that $1< x_1 <3$ and $3< x_2 <5$ were excluded. The remaining 42 data pairs also gave the fuzzy system in Equation (9), δ=1 and MSE=0.2468. In both of the above cases, the error value 0.216 still holds for the original 50 points. The use of 36 data points obtained by excluding those such that $1< x_1 <3.5$ and $2.5< x_2 <5$ did not, however, return the fuzzy system in Equation (9).

The testing of the capability of the proposed algorithm to combat noise and generalize simultaneously was done in accordance with the following procedure: The data elimination process described in the preceding paragraph was again considered and noisy data from among the 12 points indicated in Table 1 were introduced. The introduced noisy points were those with input pairs that survived elimination. Hence, 9 noisy points (set 1, 1 point from set 2 and set 3) were used among the 45 data pairs, which resulted from the first data exclusion. Also, 8 noisy points (set 1 and set 3) were used among the 42 data pairs, which remained after the second data elimination. In both cases, the algorithm returned the final fuzzy system expressed in Equation (9) with δ=1. The input-output surface is therefore as shown in Figure 7. It can be seen here that the proposed fuzzy system design algorithm is able to combat noise and generalize simultaneously.

5 Summary and Conclusions

In this study, a new algorithm for the design of Mamdani-type and simple-to-read fuzzy controllers based on a set of input-output data has been presented. It has relied on the use of a new and parametrized defuzzification strategy. The study of a property of this defuzzification method has led to the inception of a useful learning scheme whereby a consistent modification of the defuzzification parameter and rules consequents is implemented to obtain a final fuzzy system through the reduction of the data approximation error.

The algorithm has been tested through the use of a typical non-linear function, which was considered in a number of published papers. In the case of noiseless data, the data-fitting of the proposed approach has been shown better than the one pertaining to fuzzy clustering and fuzzy partition. It has been worse, however, than the data-fitting capability of the combined fuzzy-partition-neural-network, clustering-gradient-position approaches and ANFIS. Yet, the proposed algorithm gave a better representation of the shape of the non-linear function than ANFIS. When the data are noisy, a case that corresponds more to practical situations, the presented approach has given a smaller noise-free data error, better noise insensitivity and function shape representation than ANFIS. In addition, the algorithm has been shown able to extrapolate to regions of missing data and to combat noise and generalize simultaneously.

Acknowledgements

This research has been supported by the University Research Board at the American University of Beirut, Lebanon.

References

[1] K. Hayashi and A. Otsubo, “Simulator for studies of fuzzy control methods,” Fuzzy Sets and Systems, vol. 93, pp. 137-144, Jan. 1998.

[2] P. Siarry and F. Guely, “A genetic algorithm for optimizing Takagi-Sugeno fuzzy rule-bases,” Fuzzy Sets and Systems, vol. 99, pp. 37-47, 1998.

[3] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” IEEE Trans. Systems, Man and Cybernetics, vol. SMC-15, No. 1, pp. 116-132, Jan./Feb. 1985.

[4] F. Klawonn and R. Kruse, “Constructing a fuzzy controller from data,” Fuzzy Sets and Systems, vol. 85, pp. 177-193, Jan. 1997.

[5] H. Takagi and I. Hayashi, “NN driven fuzzy reasoning,” Int’l Journal of Approximate Reasoning, vol. 5, No.3. pp. 191-212, May 1991.

[6] S.Horikawa, T.Furuhashi and Y.Uchikawa, “On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm,” IEEE Trans. Neural Networks, vol. 3, No. 5 pp. 801-806, Sept. 1992.

[7] H. Nomura, I. Hayashi and N. Wakami, “ A learning method of fuzzy inference rules by descent method,” IEEE Int’l Conference on Fuzzy Systems, San Diago, California, March 18-22, 1992, pp. 203-210.

[8] J.-S.R. Jang, “ANFIS: Adaptive-network-based fuzzy inference system,” IEEE Trans. Systems, Man and Cybernetics, vol. 23, No. 3, pp. 665-685, May/June 1993.

[9] J.-S.R. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," Proceedings of the IEEE, vol. 83, No. 3, pp. 378-406, March 1995.
[10] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," IEEE Trans. Fuzzy Systems, vol. 1, No. 1, pp.7-31, Feb.1993.
[11] J.-Q. Chen, Y.-G. Xi and Z.-J.Zhang, "A clustering algorithm for fuzzy model identification," Fuzzy Sets and Systems, vol. 98, pp. 319-329, 1998.
[12] M. Delgado, A.F.G.–Skarmita and F.Martin, "A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling," IEEE Trans. Fuzzy Systems, vol. 5, No.2, pp. 223-233, May 1997.
[13] Y.Lin, G.A. Cunningham III, and S.V. Coggeshall, "Using fuzzy partitions to create fuzzy systems from input-output data and set the initial weights in a fuzzy neural network," IEEE Trans. Fuzzy Systems, vol. 5, No. 4, pp. 614-621, Nov. 1997.
[14] Y.S. Tarng, Z.M. Yeh and C.Y. Nian, "Genetic synthesis of fuzzy logic controllers in turning," Fuzzy Sets and Systems, vol. 83, pp. 301-310, Nov. 1996.
[15] B. Carse, T.C. Fogarty and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," Fuzzy Sets and Systems, vol. 80, pp. 273-293, June 1996.
[16] R. Li and Y. Zhang, "Fuzzy logic controller based on genetic algorithms," Fuzzy Sets and Systems, vol. 83, pp. 1-10, Oct. 1996.
[17] J.J. Saade, "A unifying approach to defuzzification and comparison of the outputs of fuzzy controllers," IEEE Trans. Fuzzy Systems, vol. 4, No. 3, pp. 227-237, Aug. 1996.
[18] J.V. Oliveira, "A set theoretic defuzzification method," Fuzzy Sets and Systems, vol. 76, pp. 63-71, Nov. 1995.
[19] S. Roychowdhury and B. H. Wang, "Cooperative neighbors in defuzzification," Fuzzy Sets and Systems, vol. 78, pp. 37-49, Feb.1996.
[20] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," IEEE Trans. Systems, Man and Cybernetics, vol. SMC-3, pp. 28-44, Jan.1973.
[21] J. J. Saade and H. Schwarzlander, "Ordering fuzzy sets over the real line: An approach based on decision making under uncertainty," Fuzzy Sets and Systems, vol. 50, pp. 237-246, Sept. 1992.
[22] R.E. Bellman and L.A. Zadeh, "Decision making in a fuzzy environment," Electronics Research Lab. University of California, Berkeley, Rep. No. ERL-69-8, Nov. 1969.
[23] L.A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, pp. 338-353, June 1965.
[24] J.J. Saade, "Mapping convex and normal fuzzy sets," Fuzzy Sets and Systems, vol. 81, No. 2, pp. 251-256, July 1996.