Mathware & Soft Computing 11 (2004) 143-162

Learning Fuzzy Systems – An Objective Function-Approach

F. Höppner and F. Klawonn
University of Applied Sciences Braunschweig/Wolfenbüttel
Salzdahlumer Str. 46/48
D-38302 Wolfenbüttel, Germany

Abstract

One of the most important aspects of fuzzy systems is that they are easily understandable and interpretable. This property, however, does not come for free but poses some essential constraints on the parameters of a fuzzy system (like the linguistic terms), which are sometimes overlooked when learning fuzzy system automatically from data. In this paper, an objective function-based approach to learn fuzzy systems is developed, taking these constraints explicitly into account. Starting from fuzzy c-means clustering, several modifications of the basic algorithm are proposed, affecting the shape of the membership functions, the partition of individual variables and the coupling of input space partitioning and local function approximation.

1 Introduction

There is a number of techniques that can be used to model (univariate or multivariate) input-output relationships, such as (linear) regression, fuzzy systems, neural networks, support vector machines, etc. While most of these models have well-established or even canonical methods to learn them from data, this is not the case for fuzzy systems. Quite often they are even outperformed in terms of prediction accuracy by competitive approaches. Thus, given a broad variety of high quality competitors, why should one prefer fuzzy systems? In fuzzy systems of the Takagi-Sugeno type[1] [8] the input-space is subdivided into subsets and each subset is linked to rather simple local models by means of if-then-rules. By describing the subsets variable by variable, they are easily understandable for a human who is not used to arguing in multidimensional spaces. The knowledge of a human expert, which is usually more qualitative in nature, can be cross-checked with a fuzzy model relatively simple, because the subsets are described almost in natural language (via so-called linguistic terms like "high" or "low"), the subsets tessellate

[1] We will only consider fuzzy systems of this type in this paper.

the input space along the main axes, and quite simple local models are used to express basic qualitative aspects holding in this subset. While initially the intention was to let the expert provide the rules that make up a fuzzy system, difficulties in knowledge acquisition and fusion made it more promising to let some algorithm "learn" the fuzzy systems and use the human expertise to "verify" what has been learned before it is applied in industrial applications. This is much more comfortable for the expert than fixing the numerous parameters on her or his own. Thus, if one decides to use fuzzy systems it will be for sake of simplicity, interpretability and verifiability of the resulting model, not because of prediction accuracy.

The sketched approach is, of course, only possible if the understandability and interpretability of the system is given. In a somewhat simplified view, a fuzzy system works as follows:

- The domain $X \subseteq \mathbb{R}^k$ of a (k-dimensional) input space is partitioned into c regions P_i, $\bigcup_{i=1}^{c} P_i = X$. For the moment it is sufficient to consider P_i as ordinary (rather than fuzzy) subsets of X.
- To each of the regions P_i, a local model $f_i : P_i \to \mathbb{R}$ is assigned. Given some input data $x \in X$, the system then identifies in which part of the input space it falls (that is, which element R_i of the partition is affected) and use the local model $f_i(x)$ assigned to it to predict the output value $f(x)$.

Already at this point we can outline two requirements that need to be fulfilled in order to obtain an interpretable system:

(P) Concerning the partition: The partition must be simple such that the expert can easily recall it. Two different kinds of partitions may be equally useful: Each variable can be partitioned separately and the partition of the multivariate universe is obtained as the cross-product of the univariate partitions. This makes it easier for a human to argue in multivariate spaces. Sometimes, however, variables might be that much related that a human is used to consider them in combination, as it is the case with spatial variables (combination of longitude and latitude to location or travel time and travel distance to travel speed). In both cases, the elements of the partition should be contiguous and convex.

(M) Concerning the complexity of the local models: The local models provide an answer to the question: Given the input variables lying within a limited range, what is the behavior of the output value? In order to judge about the appropriateness of such a model by a human expert, the local models should be flexible enough to express the qualitative behavior expected by the expert, but should not overcharge the expert. In many cases, a small set of simple models (such as linear or quadratic models) should be sufficient.

In fuzzy models the partitions are not crisp – the input data belongs to multiple regions R_i at the same time but to different degrees – and this is modeled by means of fuzzy membership functions. The introduction of fuzzy membership functions,

however, turns the simple application of *one local model* to predict $f(x)$ into a *mixture of models* (multiple local models have to be aggregated to calculate a single prediction $f(x)$, in TS-type systems this is done by averaging the models $f_i(x)$, weighted by the membership to the assigned partitions). This is useful near the borders of a region, because it helps to smoothly blend rather than switch abruptly from one local model to another. While the smoothness is obtained from any (continuous) set of membership functions, from the interpretability point of view it is important to really understand the membership functions as fuzzifications of a crisp partitioning of the input space. Only in this case it is guaranteed that a comparatively small number of regions attains positive membership degrees and thus only a small number of models actually influence $f(x)$. If we consider a fuzzy model using amongst others a rule "if x is approximately zero, then $f(x) = 2x+1$", we expect the resulting model to *behave near zero as it has been described*. But if another rule with model $y = -4x$ is also firing with similar strength, the resulting local model may actually be inverted. The more rules fire, the more difficult it is for the expert to cope with the mixture of local models. The interpretability of a fuzzy system relies on the fact that, in principle, the input/output behavior can be understood on a *rule by rule*-basis, which is no longer true, if arbitrary many rules fire at the same time. This poses a third requirement (specific to the fuzzification):

(F) Concerning the shape of the membership functions: Given a partition P_i (be it univariate or multivariate), the fuzzy membership functions shall be understood as fuzzifications of characteristic functions of the sets P_i, that is, the membership functions should have bounded support or decay rapidly. They should also be simple in shape and unimodal. (It would be counterintuitive if the membership of the linguistic term "fast", which is high for "200 km/h", would be higher for "230 km/h" than for "210 km/h".)

A procedure that learns a fuzzy system automatically from data must respect these requirements, otherwise the resulting system does not deserve the predicate "interpretable" (and in this a case, one may sacrifice the most important (if not the only) reason for choosing a fuzzy system). In the literature it is often argued that it is sufficient to show functional equivalence between two approaches. Being able to transfer a fuzzy system to, say, a neural network and vice versa, one could apply neural network learning techniques and use the fuzzy system representation for better readability, thereby combining the best of both worlds. This is, of course, not true, because the back-transformed neural network (after training) does not necessarily take any of the abovementioned requirements into account – and the resulting fuzzy system is therefore not interpretable (unless constraints similar to those mentioned above are explicitly formulated as constraints, see for instance [7]).

In this paper, we propose to combine techniques from (fuzzy) clustering and (fuzzy) regression to learn an interpretable fuzzy system directly from data. All of the abovementioned requirements will be addressed. The use of a clustering algorithm guarantees that the obtained membership functions can indeed be interpreted as a partitioning of the input space (P). We take care that both kinds of partitions

(axis-parallel tessellation and arbitrary partition) are possible, such that the dictionary of linguistic terms can be tailored to the experts use of the variables. Using fuzzy clustering algorithms (fuzzy c-means and derivatives) establishes the fuzzy membership functions that are characteristic for fuzzy systems. We modify existing algorithms such that requirement (F) is satisfied. Finally, we combine the clustering techniques with regression to interweave the local model fitting with the partitioning process, such that the partition is no longer determined by the distribution of the training data in input space, but actively influenced by the approximation quality of the input/output relationship.

2 Objective Function-Based Fuzzy Clustering

In this section we briefly review the fuzzy c-means [1] and related algorithms, for a thorough overview of objective function-based fuzzy clustering see [5], for instance. The objective of clustering is to assign similar data objects to the same and dissimilar data objects to different clusters. In objective function-based fuzzy clustering, every cluster is represented by a prototype p_i. Let us denote the membership degree of data object $x_j \in X$, $j \in \{1, ..., n\}$, to cluster p_i, $i \in \{1, ..., c\}$, by $u_{i,j} \in [0, 1]$. A value $u_{i,j} = 1$ means that data object x_j is unambiguously assigned to cluster p_i. Denoting the distance of a data object x_j to a cluster p_i by $d(x_j, p_i)$, the clustering process can be understood as a minimization of the objective function

$$J_m(P, U;\ X) = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{i,j}^m d^2(x_j, p_i) \tag{1}$$

If a data object has a low distance to a cluster, we can assign it to the cluster ($u_{i,j} \approx 1$), otherwise we achieve a low value of J by choosing a small membership degree ($u_{i,j} \approx 0$). The so-called "fuzzifier" m is chosen in advance and influences the fuzziness of the final partition (crisp partition as $m \to 1$ and totally fuzzy partition as $m \to \infty$; common values for m are within 1.5 and 4, 2 is most frequently used)[2]. The objective function is minimized iteratively subject to the constraints

$$\forall_{1 \le j \le n} : \sum_{i=1}^{c} u_{i,j} = 1, \quad \forall_{1 \le i \le c} : \sum_{j=1}^{n} u_{i,j} > 0 \tag{2}$$

to avoid the trivial solution (all memberships are zero). In every iteration step, minimization with respect to $u_{i,j}$ and p_i is done separately. The necessary conditions for a minimum of J yield update equations for both half-steps. Independent of the choice of the distance function and the prototypes, the membership update equation is

$$u_{i,j} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d^2(x_j, p_i)}{d^2(x_j, p_k)} \right)^{\frac{1}{m-1}}} \tag{3}$$

[2] See [6] for a motivation of the fuzzifier m and alternatives.

In the most popular fuzzy clustering algorithm, the fuzzy c-means (FCM) algorithm, the prototypes are points in X and the distance function is the Euclidean distance. Then, we obtain the optimized prototypes as the weighted mean of the data objects assigned to the cluster:

$$p_i = \frac{\sum_{j=1}^{n} u_{i,j}^m x_j}{\sum_{j=1}^{n} u_{i,j}^m.} \tag{4}$$

In this paper, we also utilize the fuzzy c-regression models (FCRM) algorithm [3], which uses polynomials as cluster prototypes. With real functions $\mathbb{R} \to \mathbb{R}$ the cluster models are characterized by the coefficients of the polynomial, that is, the prototypes are elements of $\mathbb{R}^{q+1}$ where q depends on the degree of the polynomials and the dimensionality of the input space. The Euclidean distance of FCM is replaced by the residual error $|y - h(x)|$ of a data object (x, y) (consisting of input value x and output value y) to the polynomial h. For simplicity, we consider for linear relationships extended data objects $\hat{x}$ which have an additional component $\hat{x}_0 \equiv 1$. Then, the distance function can be written as

$$d^2((x_j, y_j), p_i) \;=\; \left(y_j - p_i^\top \hat{x}_j\right)^2 \;=\; \left\| y_j - \sum_{l=0}^{k} p_{i,l} \hat{x}_{j,l} \right\|^2 .$$

For polynomials of higher degree, $\hat{x}_j$ has to be extended further, for quadratic functions and two-dimensional input $x_j = (a, b)$ we have $\hat{x}_j = (1, a, b, ab, a^2, b^2)$ such that all coefficients of the polynomial can be captured by an element of p_i. The coefficients p_i are obtained in the same fashion as the cluster centers of FCM before, we only have to replace the prototype update equation according to the modified distance function [3]:

$$p_i = \left(\sum_{j=1}^{n} u_{i,j}^m (\hat{x}_j \hat{x}_j^\top) \right)^{-1} \left(\sum_{j=1}^{n} u_{i,j}^m y_j \hat{x}_j \right) \tag{5}$$

3 Transition from Crisp to Fuzzy Partitions

3.1 A "crisp" fuzzy system

Let us revisit the "crisp notion" of a fuzzy system as it has been used in the introduction. To obtain a simple partition of a variable, say "age", it is appropriate to find a small number of characteristic values, like "about 10" (child), "about 22" (young), "about 40" (middle age) and "about 60" (old). The tuple of prototypical values $\mathbf{p} = (p_1, p_2, p_3, p_4) = (10, 22, 40, 60)$ shall be sufficient to define a crisp partitioning of the variable "age". Ignoring any other background information, the canonical approach is to define the points half-way between the prototypes as the decision boundaries for assigning a value, say 15, to its associated region, here p_1. This leads to a partition

$$\{[0, 16[, [16, 31[, [31, 50], [50, \infty]\}$$

If there is more than one input variable, we may have one partition for each variable and the regions in the input space have to address multiple variables as in

if age is *young* **and** income is *high* **then** ...

where the partition for income may be $\mathbf{q} = (10, 30, 50, 100)$ [kilo EUR]. Only if the age is young (p_2) **and** the income is high (q_3) the model in the conclusion applies (and here it does so exclusively, that is, there is no other model that applies at the same time). Continuing our interpretation of the one-dimensional case, the two-dimensional points (p_i, q_j) induce a Voronoi diagram onto the age-income plane and the abovementioned rule applies to all input data (x, y) that belong to the Voronoi cell of (p_2, q_3).

The advantage of this notion is that it holds also in case we do not have individual partitions for each of the single variables but consider some of them in combination. Then, instead of having $4 \times 4 = 16$ regularly distributed points in the plane we might have only a few irregularly distributed prototypes such as "starter" with (age,salary)=$(30, 30)$ and "experienced" with (age,salary)=$(40, 50)$ and so forth. Although the boundary between the concepts "starter" and "experienced" are no longer parallel to the main axes age and salary, the principles of subdividing the input space remain the same and correspond to the way a human would separate the concepts. Whether a partitioning along the main axes is more appropriate or not depends on the domain and the experts understanding and the fuzzy system should not force the expert to use one view or the other. Therefore it is beneficial to be able to handle both views under a unified notion.

To identify which local model applies, one has to find out in which region the input data falls. Let c points $p_i \in X$ in a (multivariate) input space be given, which may be obtained by regularly combining the prototypical points in each axis (e.g. (10,10), (10,30), (10, 50), etc.) or may represent prototypical points for joint variables (e.g. (30,40), (50,55), ...). Given a measure that yields the distance to the Voronoi cell of a prototype p_i, it can be used to identify the corresponding cell (such a measure will be developed in section 4.2). At this point, we simply assume that such a measure is given.

The prototypes p_i thus subdivide the whole input space into c Voronoi cells P_i (forming a partition of X). To each of the cells a local model f_i assigned. We introduce k minimum functions

$$\mathrm{cmin}_i : \mathbb{R}^k \to \{0, 1\} \text{ with } \sum_{i=1}^{k} \mathrm{cmin}_i(x) = 1 \text{ for any } x \in \mathbb{R}^k$$

which yield 1 if and only if the i^{th} argument is the minimum of all arguments and 0 otherwise. Only one of the $\min_i$ functions return a value of 1, if multiple arguments are minimal, as in $\min_3(2, 4, 6, 3, 2)$, ties are broken arbitrarily. Now, the input/output relationship represented by the system can be summarized concisely as

$$\sum_{i=1}^{c} \mathrm{cmin}_i(d_1(x), d_2(x), ..., d_c(x)) \cdot f_i(x)$$

where $d_i(x)$ represents the distance of x to the Voronoi cell of p_i. Since cmin_i is non-zero only in case $d_i(x)$ is minimal (and thus x belongs to Voronoi cell p_i), model $f_i(x)$ is the only local model that contributes to the output.

If we choose, for instance, linear models f_i, such a system is easily understandable and therefore well-suited for interpretation and verification by a human expert. In the next section, we will turn it into a fuzzy system.

3.2 The Fuzzification

The cmin_i functions can be understood as characteristic functions of the elements P_i of the partition induced by the prototypes p_i. We obtain a fuzzy system by generalizing these characteristic functions into fuzzy ones. This can be achieved by simply replacing the crisp cmin_i functions by fuzzy counterparts fmin_i:

$$\text{fmin}_i : \mathbb{R}^k \to [0,1] \text{ with } \sum_{i=1}^{k} \text{fmin}_i(x) = 1 \text{ for any } x \in \mathbb{R}^k$$

The fuzzified minimum function can be motivated as follows: In case of the four values $\mathbf{d} = (1, 100, 150, 99)$ we have no problems with clearly identifying the value 1 as the minimum. But in case of $\mathbf{d'} = (1, 100, 150, 2)$ things are a little less certain. Still, 1 is the minimum, but given the other values, 2 is also very close to the minimal value of 1. In a fuzzy fashion, we could say that 1 is the minimum value to some degree, say 0.6, and 2 to some (smaller) degree, say 0.4. In consequence, when considering $(1+\varepsilon, 100, 150, 2-\varepsilon)$ for any ε we want to have a smooth transition of the minimality degree fmin_1 from 1 to 0 (and fmin_4 from 0 to 1) as ε goes from $-\infty$ to ∞, with $\text{fmin}_1 = \text{fmin}_4$ for $\varepsilon = 0.5$. Basically we want to turn the discontinuous cmin_i functions into continuous ones.

There are many possibilities to define such minimum functions, we propose to use the following here:

Theorem 1 (Fuzzified Minimum Function) *Let $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a strictly increasing function with $f(x) \geq x$, let $\eta \in \mathbb{R}_{\geq 0}$. With $d = (d_1, ..., d_k) \in \mathbb{R}^k$, $D_s = (f(d_s - \min\{d_1, .., d_k\}) + \eta)^q$, $q \geq 1$, we define*

$$\text{fmin}_s(d) = \left(\sum_{i=1}^{k} \frac{D_s}{D_i} \right)^{-1} \tag{6}$$

Then, the following inequality holds:

$$\left| \sum_{s=1}^{k} \text{fmin}_s(d) \cdot d_s - \min\{d_1, d_2, ..., d_k\} \right| < \eta^q r + \eta(k - r - 1) \leq \eta(k-1)$$

where r is the number of indices s for which d_s has at least a distance of $1-\eta$ from the minimum: $r = |\{s \,|\, 1 \leq s \leq k,\ d_s - \min\{d_1, d_2, ..., d_k\} > 1 - \eta\}|$

Figure 1 shows an example where we take the pointwise minimum of three functions (dashed lines). The resulting fuzzified minimum is displayed for two different values of $\eta = 0.1/0.2$ (solid lines) using $q = 1.5$. According to the theorem, the error is bounded by 0.06/0.18 if the minimum is clearly separated from the other values.

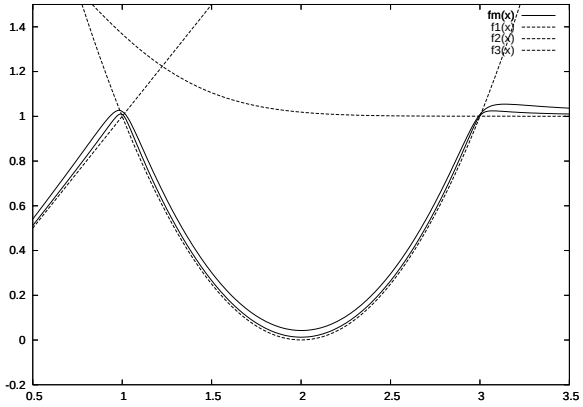


Figure 1: Minimum of three functions.

The fuzzy minimum functions have several desirable properties: Despite the occurrence of the minimum expression, they are differentiable if $q > 1$ and f is differentiable. The functions actually represent an approximation of the crisp minimum function ($\min\{d_1, d_2, ..., d_k\} = \sum_{i=1}^{k} \text{cmin}_i(d)$) and by variation of the parameter η the approximation quality can be controlled.

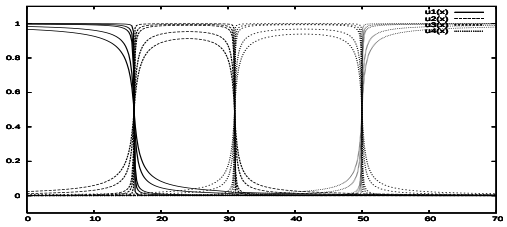
Obviously, the effect of replacing the crisp with the fuzzy minimum function does not destroy any of the "partitional properties", because the fuzzification represents only a smoothing operation. Therefore requirement (P) still holds. The membership functions do not show clear local extrema (unimodality) and decay rapidly outside the Voronoi cells. Therefore, requirement (F) is also fulfilled.

In the next sections, we investigate how to modify and extend existing clustering and regression algorithms such that they provide a method to learn fuzzy system of this kind automatically from data.

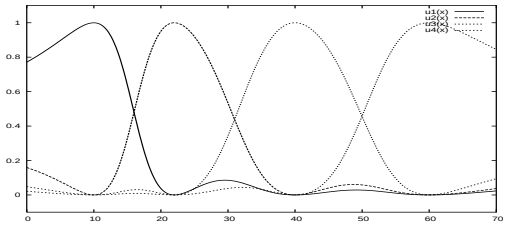
4 An Objective Function-based Fuzzy System

Objective function-based fuzzy clustering algorithms, as they have been discussed briefly in section 2, will provide the basis for the construction of an objective function, whose minimum represents a fuzzy system that models the input/output-relationship present in the data. Fuzzy clustering algorithms have been used before to learn fuzzy systems from data, and it has been outlined several times that the membership functions of the fuzzy c-means algorithm are not very well suited to be used in a fuzzy system. Figure 2(b) shows the membership functions of FCM in

case of the age partition from the previous section (for $m = 2$). The leftmost (resp. rightmost) membership function should have constantly high values to the left (resp. right) and the local maxima are also counterintuitive. Thus, the functions exhibit a number of undesired properties which are in contrast to requirement (F). The problem of unimodality can be solved by using possibilistic memberships [2], but the possibilistic c-means is not a partitional but a mode-seeking algorithm.



(a) Fuzzified crisp partitions.



(b) FCM membership functions ($m = 2.0$).

Figure 2: Different kinds of membership functions.

In the following subsections, we will modify the objective function of the fuzzy c-means algorithm such that (a) the notion of distance corresponds to the understanding of fuzzy systems as described in section 3.1, (b) the membership functions satisfy condition (F), (c) the expert can choose what kind of partitioning she or he wants to use for which variables (condition (P)), and (d) the local models and the partition are developed simultaneously.

4.1 Membership Functions as Fuzzified Minimum Functions

The membership functions (3) obtained from the fuzzy c-means algorithm are already very similar to the fuzzified minimum function (6): if we set $\eta = 0$, $f(x) = x^m$, $q = 1/(m-1)$ the only difference is the additional occurrence of the minimum term in (6). Since we want the FCM algorithm to provide (6) as membership functions, we have to incorporate the minimum term somehow. Formally, this is quite difficult, because the update equations evolve from differentiating an objective function and solving for its parameters. Solving for the prototype parameters, for instance, becomes difficult since they occur within a minimum expression. Therefore, we circumvent the problem by introducing another set of parameters a_j,

one for each data object, and define a new distance measure d' as being the old distance d reduced by a_j under the condition that a_j is the minimum of all distance values d. We then "optimize" the a_j values in a third stage of the alternating optimization:

1. calculate the membership degrees (assuming prototypes and a_j as being constant)
2. calculate the prototypes (assuming memberships and a_j as being constant)
3. calculate the a_j (assuming prototypes and memberships as being constant)

By doing so, we arrive at identical update equations for the membership and prototype update (only the reduction by a_j appears in the membership calculation due to the usage of the distance d' instead of d).

At this point, we have membership degrees $u_{i,j}$ which can be interpreted as fuzzified minimum degrees (6) if we set $f(x) = x^m$, $q = 1/(m-1)$ and require $a_j = \min_i d_i - \eta$ for some constant $\eta > 0$.

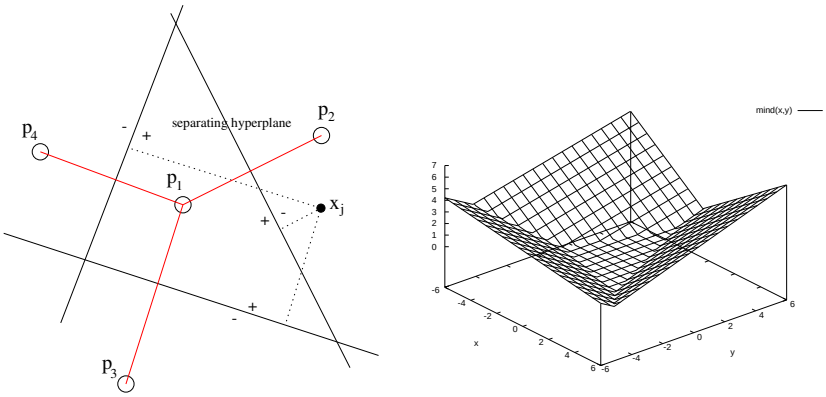
4.2 Memberships Induced by Voronoi Distance

In the fuzzy c-means algorithm, the prototypes are points p_i in the input space. In section 3.1 we have discussed a system where the prototypical values p_i induce Voronoi cells and that the belongingness to these cells decides which local model has to be applied. As already mentioned, the Euclidean distance of a data object x_j to the hyperplane that separates the points p_i and p_s is given by $|(x_j - h_{s,i})^\top n_{s,i}|$ where $h_{s,i}$ is a point on the hyperplane, e.g., $h_{s,i} = (p_s + p_i)/2$, and $n_{s,i}$ is the normal vector $n_{s,i} = \beta_{s,i} \cdot (p_i - p_s)$ with $\beta_{s,i} = \frac{1}{\|p_i - p_s\|}$ for $s \neq i$. If we do not take absolute values, we obtain *directed* distances $(x_j - h_{s,i})^\top n_{s,i}$, which become positive if x_j lies on the same side as the cluster center and negative if x_j lies on the opposite side. Taking the absolute value of the minimum over all the directed distances yields the distance to the *border of the cell*. If x_j lies within the Voronoi cell of cluster i, then the distance to the *cell* is zero. We can formalize this special case easily by setting $\beta_s = 1$ and defining:

$$d_V(x_j, p_i) = \left| \min_{1 \leq s \leq c} (x_j - h_{s,i})^\top n_{s,i} \right|$$

In Fig. 3(a), x_j is closest to the separating line between p_1 and p_2, therefore this distance serves as the distance to the Voronoi cell of p_1. The graph of d_V for the 4 clusters of Fig. 3(a) is shown in Fig. 3(b).

If we do not scale the normal vectors n_s to unit length, but assume $\beta_s = 1$ for all s, we preserve the shape of d_V (position of hyperplanes does not change), only the gradient of the different hyperplanes varies. The following lemma establishes a link between the Voronoi distance to the cell induced by p_i and the fuzzified minimum functions (6) via this *scaled Voronoi distance*:



(a) Voronoi cell of centroid p_1.

(b) Distance to Voronoi cell.

Figure 3: Developing a distance between a point and a Voronoi cell.

Lemma 1 *Given a Voronoi diagram induced by a set of distinct points p_i, $1 \leq i \leq c$, and a point x. Using $\beta_{s,i} = 1$ for all $1 \leq s, i \leq c$, the (scaled) distance between x and the Voronoi cell of point p_i is given by*

$$d_V(x, p_i) = \frac{1}{2}\left(\|x - p_i\|^2 - \min_{1 \leq s \leq c} \|x - p_s\|^2\right) \tag{7}$$

This gives a very nice interpretation of the fuzzy c-means algorithm using distance values reduced by the minimum of all distances as described in the last section: Establishing the fuzzified minimum functions as membership functions is equivalent to providing the Voronoi distance to the cell of prototype p_i instead of the Euclidean distance to p_i (as in standard FCM). This corresponds well to our understanding of a fuzzy system in section 3.1.

Note that with FCM squared Euclidean distances are used to determine the membership degrees, but if we use the Voronoi distance we use Euclidean distances to the Voronoi cell, which are not squared. Therefore, the modification may reduce the sensitivity to noise and outliers.

4.3 Effect on the Membership Degrees

The proposed modifications of FCM yield different membership functions than the original FCM algorithm. At the beginning of this section we have mentioned that the membership functions of the original algorithm are not very well suited for usage in a fuzzy system (constraint (F) violated). As an example, consider the one-dimensional input space for the variable *age* (cf. figure 2(b)). The prototype with the biggest number is $p_4 = 60$. When a data object is given by $x = p_4 + \epsilon$, the membership degrees (of original FCM) tend to approach $\frac{1}{c}$ for all clusters as $\epsilon \to \infty$! We would expect, of course, that $u_4 \to 1$ while $u_i \to 0$ for all other clusters. What is the reason for this undesired behavior?

With FCM the relative distances (cf. (3)) define the degree of membership to a cluster, e.g., if the distance between x_j and p_1 is half the distance to p_2, the membership degree $u_{1,j}$ is twice as large as $u_{2,j}$. If we consider crisp membership degrees things are different, the membership degree does not depend on the ratio of distances, but the distances serve as threshold values. If the distance to p_1 is smaller than to p_2 – no matter how much smaller – we always have $u_{j,1} = 1$.

Let us now consider the case of fuzzified minimum functions (6) again and assume that p_i is closest to x_j. No matter if x_j is far away from p_i (but all other p_k are even further away) or x_j is very close to p_i, the numerator of the distance ratio is always constant η. Inside the Voronoi cell of p_i, the distance to cluster i is considered to be constant η. The membership degrees $u_{k,j}$ are therefore determined by the denominator, that is, mainly by $d^2(x_j, p_k)$. Therefore, the membership degrees obtained by (6) are no longer defined by a ratio of distances, but the distances have the flavor of a threshold value (cf. figure 2(a)).

Surprisingly, besides the different shape of the membership functions, the resulting algorithm performs very similar to conventional FCM in terms of resulting cluster centers. Concerning the membership functions, however, the modified FCM is much closer to its crisp original, the ISODATA or k-means algorithm.

4.4 Regularly distributed prototypes

Quite often the variables in a fuzzy system are partitioned individually and the multidimensional region, for which a certain model holds, is specified by a conjunction of conditions on the individual variables (as in "**if** age is **young** and income is **high** ..."). For reasons of consistency and interpretability, the membership function that is referred to by the linguistic term **young** must be identical in all rules in which it occurs. That is, the prototypical value for **young** is not chosen with respect to this single rule (and its model), but to all rules in which this term is used. In our approach, the linguistic terms are represented by prototypes p_i in the *multidimensional* input space, that is, a prototype represents multiple linguistic terms at the same time (one for each variable). The prototypes $(age_1, income_1)$ for the rule "**if** age is **young** and income is **medium**..." and $(age_2, income_2)$ for the rule "**if** age is **young** and income is **high**..." must always share the same value for the age component, that is, $age_1 = age_2$. Similarly, for the prototype $(age_3, income_3)$ of the rule "**if** age is **old** and income is **high**..." we have $income_3 = income_2$. That is, we have to respect the regular distribution of the prototypes on a multidimensional grid.

Standard FCM distributes the prototypes irregularly in the input space. If the expert wants to use one partition per variable, the fuzzy c-means algorithm must be modified to guarantee that all prototypes represent jointly a regular grid in the input space. Given k input variables, suppose we want to divide the domain of variable v_i into N_i linguistic terms, induced by points $p_{i,j}$, $i \in \{1, ..., k\}$, $j \in \{1, ..., N_i\}$. Then, we have a tessellation of X into $\prod_{i=1}^{k} N_i$ regions. We denote any

region by the tuple of indices $(i_1, i_2, ..., i_k) \in I$,

$$I = \{(i_1, i_2, ..., i_k) \mid i_j \in \{1, .., N_i\}, j \in \{1, .., k\}\}.$$

Each tuple addresses the subset of the input space for which a rule's premise would read like

if v_1 is p_{1,i_1} and v_2 is p_{2,i_2} and ... and v_k is p_{k,i_k} **then** ...

The prototype that represents the region addressed in the premise is

$$\begin{pmatrix} p_{1,i_1} \\ p_{2,i_2} \\ \vdots \\ p_{k,i_k} \end{pmatrix}$$

The set of c prototypes, $c = \prod_{i=1}^{k} N_i$, is given by

$$P = \{(p_{1,i_1}, p_{2,i_2}, \cdots, p_{k,i_k}) \mid (i_1, i_2, ..., i_k) \in I\}$$

Given these definitions, the objective function turns into:

$$J = \sum_{j=1}^{n} \sum_{(i_1,i_2,...,i_k)\in I} u^m_{(i_1,i_2,...,i_k),j} \begin{pmatrix} x_1 - p_{1,i_1} \\ x_2 - p_{2,i_2} \\ \vdots \\ x_k - p_{k,i_k} \end{pmatrix}^2 \tag{8}$$

Contrary to standard fuzzy c-means, where each prototype is determined separately, now we have to determine the $p_{i,j}$ separately:

Lemma 2 *The necessary conditions for a minimum of the objective function (8) under constraints (2) are given by:*

$$p_{l,r} = \frac{\sum_{j=1}^{n} \sum_{(i_1,i_2,...,i_k)\in I, i_l=r} u^m_{(i_1,i_2,...,i_k),j} \cdot x_l}{\sum_{j=1}^{n} \sum_{(i_1,i_2,...,i_k)\in I, i_l=r} u^m_{(i_1,i_2,...,i_k),j}} \tag{9}$$

Since the derivation of the necessary condition arrives at a unique solution, convergence of the iterative alternating optimization scheme is guaranteed [4].

It is also possible to provide a "mixed mode" fuzzy c-means variant, where the user specifies a partitioning of the variables together with the number of prototypes for each element of the partition. For instance the input space $(age, income, longitude, latitude)$ could be partitioned into $\{\{age\}, \{income\}, \{longitude, latitude\}\}$. For the multivariable set $\{longitude, latitude\}$ the equation (9) must then be interpreted as a vector rather than as scalar equation.

4.5 Combining Clustering and Regression

For each cluster or region in the input space, a local model has to be identified that fits the data within the region best. For each cluster, we may instantiate a

polynomial of low degree (1 or 2, for instance) to locally approximate the input-output relationship in this cluster, as it is done with switching regression models (cf. section 2 and [3]). Performing the clustering first, before we then fit the local models to the identified clusters, is not a good solution, because then the output values have no influence on the partitioning. Consider two data sets with regularly distributed data points in the input space. Both data sets are identical with respect to the input variables, however, the output values may represent completely different functions. If we decouple clustering and model fitting, we would come up with identical partitions for both functions. This may not be a problem if the capabilities of the local model are universal in the sense that they can represent any function. In our case of fuzzy systems, however, we have decided to choose rather simple linear or quadratic functions. So, the modeling capabilities are limited and the tessellation of the input space should be chosen in a way that supports the approximation by simple local models.

Furthermore, in real world data it is likely that regions of high data density simply indicate the operating points of the systems and not necessarily good centers for local models. To achieve a better interaction between partitioning and model fitting we interweave both steps and execute them simultaneously, such that bad regression fits can have influence on the partitioning and therefore lead to a better fit next time. Since both algorithms (fuzzy clustering and switching regression models) are objective function-based, their combination is straightforward. We simply use the sum of both distance functions (FCM and FCRM) in the modified clustering algorithm:

$$d^2(\,(x_j,y_j)\,,\,(p_i,q_i)\,) \;=\; \underbrace{\|x_j-p_i\|^2}_{\text{FCM distance}} \;+\; \underbrace{\left(y_j-q_i^\top \hat{x}_j\right)^2}_{\text{FCRM distance}}\,. \tag{10}$$

The FCM distances are taken with respect to the input value x_j and cluster center p_i, while the FCRM distances are taken with respect to the given output value y_j and the value of the polynomial at $\hat{x_j}$ with coefficients q_i.

Since there are no dependencies between the parameters of the modified clustering and regression prototypes (p_i and q_i), the same prototype update equations hold for the combined algorithm. Nevertheless, cluster centers and polynomials influence each other indirectly by means of the membership degrees, which depend on the distance to both models.

5 Examples

In this section we give a few examples for learning a one-dimensional function over a two-dimensional input space. We use very simple linear local models and partition the two input variables into 3 to 4 elements only. It is obvious that the mean square error may become quite large locally. However, as mentioned in the introduction, our aim is not to get a close quantitative approximation, but to capture the qualitative aspects of the functions.

Figure	partition x-axis	partition y-axis
4	[0,0.6[, [0.6,1.2[, [1.2,2.5[, [2.5,4.0]	[0,1.0[, [1.0,2.0[, [2.0,3.0]
5	[0,1.0[, [1.0,1.75[, [1.75,2.5]	[0,1.25[, [1.25,2.25[, [2.25,3.0]
6	[0,0.85[, [0.85, 1.65[, [1.65,2.35[, [2.35,3.0]	[0,1.1[, [1.1,2.1[, [2.1,3.0[

Table 1: Partitions obtained for figures 4-6.

The figures 4-6 show the functions $5 \cdot \exp(-5 \cdot (x-1)^2) + \exp(x/2)$, $x \cdot y \cdot \cos(x^2 \cdot y/2)$ and $\sin(x^2) \cdot \cos(y^2)$, resp. We sampled the input space uniformly and complemented the points with the output values. The location of the data points in the input space therefore give no hints about an appropriate partition of the input variables. The figures show the resulting models with the original functions superimposed. One can easily distinguish the local regions in which the local models apply. The partitions of the x and y variable have been adjusted automatically (cf. table 1) to improve the fit of the local linear models, the major qualitative aspects of the functions have been captured. For instance, in figure 4, the function does not depend on variable y, leading to a uniform distribution of the prototypes in this variable. For the variable x, however, the prototypes deviate from the uniform distribution in order to better approximate the function with local linear models.

If we would have allowed for complex models (e.g. polynomials of high degree), the approximation error would be smaller, but more difficult to understand by the expert – and the partitioning of the input space would not be that meaningful in itself, because the models are capable of approximating almost any part of the function equally well and therefore would not provide substantial feedback for the partitioning of the input space.

6 Conclusions

In knowledge discovery applications, one tries to discover understandable and potentially useful models from historical data. Fuzzy systems can be useful in this case, if one succeeds in learning them automatically while at the same preserving their interpretability. In this paper, we have outlined some fundamental properties a fuzzy system should have in order to be interpretable. We discussed a "non-fuzzy reference system" with clear and simple semantics, which is then turned into a fuzzy system by simply fuzzifying a minimum function used in the non-fuzzy counterpart. On the basis of fuzzy clustering and regression techniques, we developed an objective function-based algorithm which follows the clear semantic of the reference system and also optimizes the partitions of the input variables to improve the overall fit of the local models.

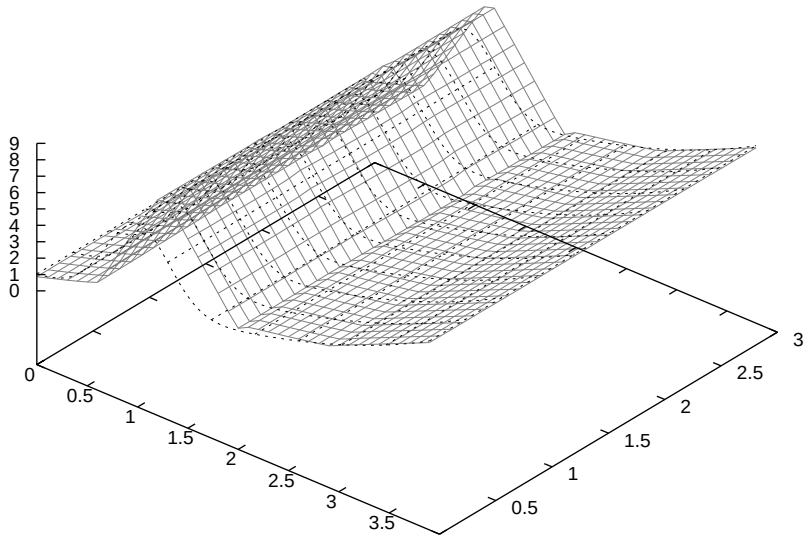


Figure 4: Example 3

A Proofs

Proof of Lemma 1: Some simple transformations yield the following chain of equalities

$$
\begin{aligned}
& d_V(x,p_i) \\
= \;& \left| \min_{1\le s\le c} \left(x - \frac{p_s+p_i}{2} \right)^\top (p_i - p_s) \right| \\
= \;& \left| \min_{1\le s\le c} x^\top (p_i - p_s) - \frac{1}{2}(p_i^\top p_i - p_s^\top p_s) \right| \\
= \;& \frac{1}{2} \Big| \min_{1\le s\le c} x^\top x - 2x^\top p_s + p_s^\top p_s - \\
& \qquad + (x^\top x - 2x^\top p_i + p_i^\top p_i) \Big| \\
= \;& \frac{1}{2} \left| \min_{1\le s\le c} \|x-p_s\|^2 - \|x-p_i\|^2 \right| \\
\overset{(\star)}{=} \;& \frac{1}{2} \left(\|x-p_i\|^2 - \min_{1\le s\le c} \|x-p_s\|^2 \right)
\end{aligned}
$$

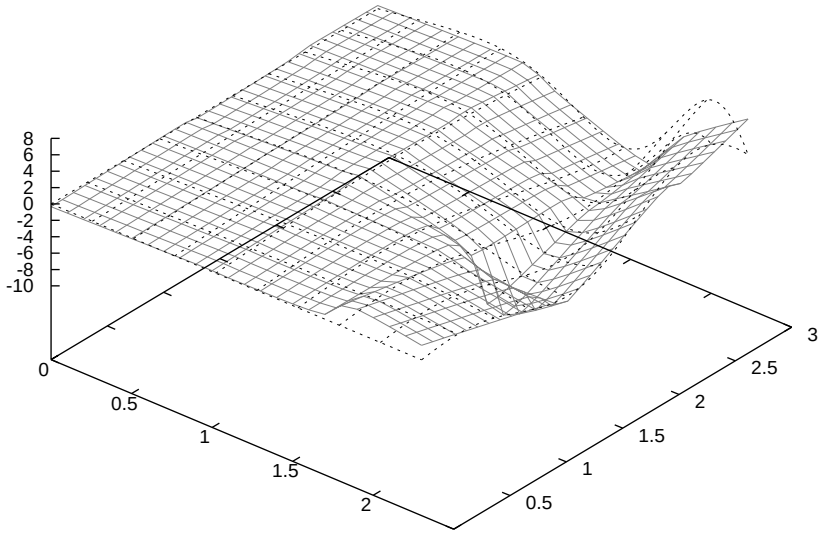


Figure 5: Example 2

In equation ($\star$) we have used the trivial fact that any $\|x - p_i\|$ is greater than or equal to $\min_{1 \leq s \leq c} \|x - p_s\|$. ■

Proof of Theorem 1: We have the following equality

$$\begin{aligned}
& \sum_{s=1}^{k} \frac{d_s}{D_s \sum_{i=1}^{k} \frac{1}{D_i}} \\
= & \sum_{s=1}^{k} \frac{d_s}{D_s \frac{\sum_{i=1}^{k} \prod_{t=1, t \neq i}^{k} D_t}{\prod_{i=1}^{k} D_i}} \\
= & \sum_{s=1}^{k} \frac{d_s \prod_{i=1}^{k} D_i}{D_s \sum_{i=1}^{k} \prod_{t=1, t \neq i}^{k} D_t} \\
= & \sum_{s=1}^{k} \frac{d_s \prod_{i=1, i \neq s}^{k} D_i}{\sum_{i=1}^{k} \prod_{t=1, t \neq i}^{k} D_t} \\
= & \frac{\sum_{s=1}^{k} d_s \prod_{i=1, i \neq s}^{k} D_i}{\sum_{i=1}^{k} \prod_{t=1, t \neq i}^{k} D_t} \qquad (11)
\end{aligned}$$

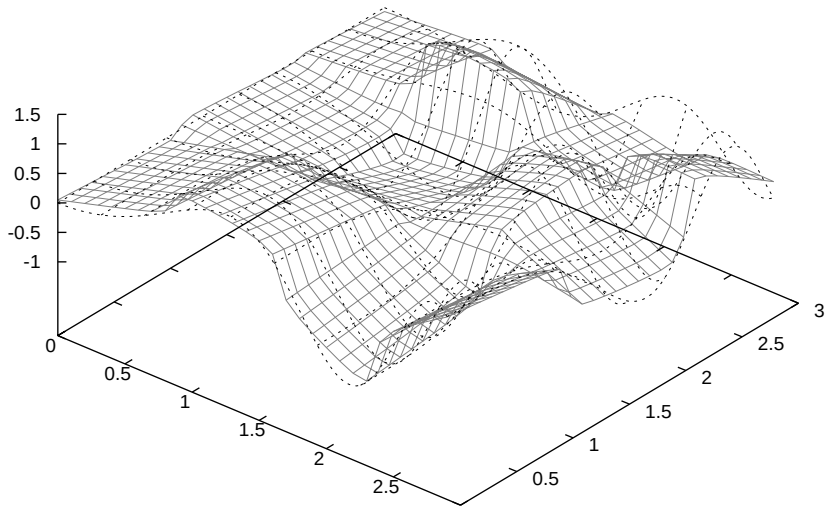


Figure 6: Example 1

Using the abbreviations $M = \min\{d_1, d_2, ..., d_k\}$ we estimate the approximation error as follows

$$
\begin{aligned}
& \left| \frac{\sum_{s=1}^{k} d_s \prod_{i=1,i\neq s}^{k} D_i}{\sum_{s=1}^{k} \prod_{i=1,i\neq s}^{k} D_i} - M \right| \\
= & \left| \frac{\left(\sum_{s=1}^{k} d_s \prod_{i=1,i\neq s}^{k} D_i\right) - M\left(\sum_{s=1}^{k} \prod_{i=1,i\neq s}^{k} D_i\right)}{\sum_{s=1}^{k} \prod_{i=1,i\neq s}^{k} D_i} \right| \\
= & \left| \frac{\sum_{s=1}^{k} (d_s - M) \prod_{i=1,i\neq s}^{k} D_i}{\sum_{s=1}^{k} \prod_{i=1,i\neq s}^{k} D_i} \right| \\
\overset{\star^1}{=} & \left| \frac{\sum_{s=2}^{k} (d_s - M) \prod_{i=1,i\neq s}^{k} D_i}{\sum_{s=1}^{k} \prod_{i=1,i\neq s}^{k} D_i} \right| \\
\overset{\star^2}{\leq} & \left| \frac{\eta^q \sum_{s=2}^{k} (d_s - M) \prod_{i=2,i\neq s}^{k} D_i}{\sum_{s=1}^{k} \prod_{i=1,i\neq s}^{k} D_i} \right|
\end{aligned}
$$

$$
\begin{aligned}
&\overset{\star^3}{<} && \left| \frac{\eta^q \sum_{s=2}^{k} (d_s - M) \prod_{i=2, i\neq s}^{k} D_i}{\prod_{i=2}^{k} D_i} \right| \\
&= && \left| \eta^q \sum_{s=2}^{k} \frac{(d_s - M)}{D_s} \right| \\
&\leq && \eta^q \sum_{s=2}^{k} \left| \frac{(d_s - M)}{D_s} \right| \\
&\overset{\star^4}{<} && \eta^q \sum_{s=2}^{k} \left| \frac{(d_s - M)}{(d_s - M)(d_s - M + \eta)^{q-1}} \right| \\
&= && \eta^q \sum_{s=2}^{k} \left| \frac{1}{(d_s - M + \eta)^{q-1}} \right| \\
&\overset{\star^5}{\leq} && \eta^q \sum_{s=2}^{k} \left| \frac{1}{\eta^{q-1}} \right| \\
&= && \eta(k-1)
\end{aligned}
$$

Remarks:

- $\star^1$ Without loss of generality we have assume that d_1 is the minimum and have $(d_1 - M) = 0$.
- $\star^2$ From $d_1 = M$ we can conclude $D_1 = (f(d_1 - d_1) + \eta)^q \leq \eta^q$.
- $\star^3$ We have dropped all summands in the denominator $\sum_{s=1}^{k} \prod_{i=1, i\neq s}^{k} D_i$ that contain D_1. All summands are positive.
- $\star^4$ We drop one η in the denominator $D_s = (d_s - M + \eta)(d_s - M + \eta)^{q-1}$ which makes the term smaller.
- $\star^5$ Here we assume the worst case that all d_s are minimal and thus $d_s - M = 0$. (However, if this would actually be the case, we can see from the equality $\star^1$ that the approximation error is zero.) We also obtain an equality if $q = 1$.

If some d_s, $s \in \{2, 3, \ldots, k\}$, have reached a distance $d_s - M \geq 1 - \eta$ from the minimum, the estimation can be improved. If we continue from the result after $\star^3$ we have $d_s - M < f(d_s - M) + \eta < (f(d_s - M) + \eta)^q = D_s$ and thus may substitute $(d_s - M)$ by D_s. This leads us to an error below $\eta^q(k-1)$.

To summarize both estimations, if there are r values that have a distance of at least $d_s > 1 - \eta + M$, we have an error smaller than $\eta(k - r - 1) + \eta^q r$. ■

Proof of Lemma 2: Equation (9) is easily obtained by setting $\frac{\partial J}{\partial p_{l,r}} = 0$ and solving for $p_{l,r}$. ■

References

[1] J. C. Bezdek. A convergence theorem for the fuzzy ISODATA clustering algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(1):1–8, Jan. 1980.

[2] R. N. Davé and R. Krishnapuram. Robust clustering methods: A unified view. *IEEE Trans. on Fuzzy Systems*, 5(2):270–293, May 1997.

[3] R. J. Hathaway and J. C. Bezdek. Switching regression models and fuzzy clustering. *IEEE Trans. on Fuzzy Systems*, 1(3):195–204, Aug. 1993.

[4] F. Höppner and F. Klawonn. A contribution to convergence theory of fuzzy c-means and derivatives. *IEEE Trans. on Fuzzy Systems*, 11(5):682–694, Oct. 2003.

[5] F. Höppner, F. Klawonn, R. Kruse, and T. A. Runkler. *Fuzzy Cluster Analysis*. John Wiley & Sons, Chichester, England, 1999.

[6] F. Klawonn and F. Höppner. What is fuzzy about fuzzy clustering? – understanding and improving the concept of the fuzzifier. In *Advances in Intelligent Data Analysis*, pages 254–264. Springer, 2003.

[7] J. V. d. Oliveira. Semantic constraints for membership function optimization. *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, 29(1):128–138, Jan. 1999.

[8] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. on Systems, Man, and Cybernetics*, 15:116–132, 1985.