Fuzzy Modeling with Population-based Optimization: Design and Analysis

by

Ali Safari Mamaghani

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in
Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Fuzzy Rule-based Systems (FRBS) form a commonly encountered category of fuzzy models and play an essential role by developing a human-centric modeling framework. Fuzzy rule-based modeling aims at identifying the structure and the parameters of "if-then" fuzzy rules so that a desired input/output mapping is achieved. One of the most widely-used architectures in fuzzy modeling come in the form of Takagi–Sugeno–Kang (TSK) rules ("if-then" rules), which use linear or, more generally, local polynomial functions in the conclusions (consequences) of the rules, rather than linguistic terms (fuzzy sets).

This dissertation's primary aim is concerned with the structural identification and parametric optimization of data-driven fuzzy models using some population-based optimization methods. Three different architecture are introduced and studied.

In the first architecture, the TSK model's structural optimization is achieved by the particle swarm optimization (PSO), with the intent of its complexity management. This is accomplished in two ways: (i) by structuralization of the antecedents and (ii) by structuralization of the consequences of the fuzzy rules. More specifically, this study contributes to the complexity management of fuzzy models by focusing on (i) the efficient arrangement (reduction) of the input spaces over which the antecedents of rules are formed and (ii) allocating the orders of local polynomial functions across the consequences of the rules. This architecture's originality comes with the flexibility of FRBS that is endowed by admitting variability of input spaces standing in the antecedents of different rules and the variability of orders of polynomials (local functions) forming the consequences of the rules. PSO as an optimization vehicle is guided by the root mean squared error (RMSE) accuracy criterion to realize the efficient arrangement of input spaces and

an allocation of the orders of the individual polynomials. In this optimization process, the Fuzzy C-Means (FCM) algorithm is employed to create fuzzy sets in the rules' antecedents, while the standard Least Square Error (LSE) optimization criterion is minimized to determine the coefficients of the polynomials in the consequences. The proposed model's performance is quantified using numeric data, including both synthetic and machine learning datasets.

The second architecture in this dissertation is realized with the aid of genetic programming (GP). The proposed architecture employs GP to form fuzzy logic expressions involving logic operators and information granules (fuzzy sets) located in the input space, and used to predict information granules in the output space. We propose an architecture realizing logic processing, with the structural optimization of the model accomplished by multi-tree genetic programming and the parametric optimization completed by gradient-based learning. The granulation of information used in this architecture is developed using the FCM clustering algorithm. The novelty of this study is two-fold: (i) it comes with the flexibility of the logic-oriented structure of fuzzy models, and (ii) our architecture is designed to handle high-dimensional data by alleviating the detrimental effect of distance concentration hampering the effectiveness of standard TSK FRBS. The study is illustrated through some experiments that give a detailed insight into the fuzzy models' performance. A comprehensive comparative analysis is also covered.

The third architecture in this dissertation aimed at the generalization of the TSK FRBS. In the current literature, most of the TSK models with polynomial (linear) consequences have been studied; however, the TSK models' design with non-linear consequences has not been discussed to a great extent. The originality of the introduced architecture comes with the TSK model's generalization, which employs a family of non-linear and linear local models rather than only linear models forming the rules' consequences. The proposed modification reduces the model's

complexity (number of rules) while preserving the desired accuracy. The introduced architecture benefits from PSO and LSE to extract the consequences, whereas fuzzy sets standing in the antecedents of rules are formed by the FCM clustering algorithm.

Preface

The research presented in this dissertation was performed under the supervision of Dr. Witold Pedrycz and was supported by funding from the National Sciences and Engineering Research Council of Canada (NSERC).

Chapter 3 of this dissertation includes the materials published as A. S. Mamaghani and W. Pedrycz, "Structural Optimization of Fuzzy Rule-based Models: Towards Efficient Complexity Management," Expert Systems with Applications 152, pp. 1-12, 2019. I was responsible for the manuscript composition, idea development, and experimental process, including programming and experimental execution and analysis. Dr. Pedrycz was involved in the concept formation and was helpful in guiding me through the existing literature on the topic and supervising manuscript composition and experimental structure.

Chapter 4 of this dissertation includes the materials published as A. S. Mamaghani and W. Pedrycz, "Genetic-Programming Based Architecture of Fuzzy Modeling: Towards Coping with High Dimensional Data," IEEE Transactions on Fuzzy Systems, pp.1-11, 2020 (Early access). I was responsible for the idea development, the implementation of experimental code, data collection, result analysis, and manuscript composition of this study. Dr. Pedrycz worked on this study in a supervisory role, aiding in the initial formation of the research topic and having meaningful contributions to the manuscript composition and refinement process.

Chapter 6 of this dissertation includes the material is to be submitted to IEEE Transactions on Fuzzy Systems as A. S. Mamaghani and W. Pedrycz, “A PSO-aided Development of TSK Fuzzy Models Employing Various Non-linear Consequences.” Dr. Pedrycz was instrumental in guiding

me through the creation of this study, contributed significantly to idea development, and helped with my learning process regarding manuscript composition. A significant amount of the idea development, all program implementation and data collection, and result analysis was performed by me under his supervision.

Dedication

The time we had dad,

wasn't nearly enough

to pack in an entire

lifetime of love.

This work is dedicated to the memory of my beloved father

and

to all victims of Ukraine flight PS752.

Acknowledgements

Special appreciation goes to my supervisor, Dr. Witold Pedrycz, for his valuable suggestions and guidance in my research direction, generous sharing of his professional and personal experiences.

I am grateful to my examination committee, Dr. Marek Reformat, and Dr. Petr Musilek, Dr. Ergun Kuru, Dr. Yiyu Yao, and Dr. Venkata Dinavahi for taking their precious time to serve on my committee.

I am indebted to my family members especially, my dear mother, Robab Badamchi, for her dedication and encouragement throughout my life. Last but not least, I would like to thank my lovely wife, Elnaz Aliasl, and of course my sweet daughter Elay for their understanding, constant support, and love during this research period. This accomplishment would not have been possible without them.

Finally, a heartfelt thank you to all Electrical and Computer Engineering departmental staff, who have always been friendly and helpful. Thank you specifically to Pinder Bains and Wendy Barton, who both provided exceptional student support during my time here.

Ali Safari Mamaghani

University of Alberta

July 2021

Table of Contents

List of Tables

List of Figures

List of Symbols

X	Data set
$\boldsymbol{x}_k$	k^{th} data point in Data set X
$target_k$	Target of k^{th} data point
N	Number of data points
n	Number of features
R^n	n dimensional space
U	Partition matrix
u_{ik}	ik^{th} element of partition matrix U
V	Prototype matrix
σ_j	Standard deviation of the j^{th} feature
c	Number of clusters
m	Fuzzification coefficient
y_k	k^{th} output data
$\boldsymbol{v}_i$	i^{th} prototype in input space
w_i	i^{th} prototype in output space
R_i	i^{th} fuzzy rule
A_i	Membership function of the i^{th} fuzzy set in the input space
$\hat{y}$	Output of model
$\boldsymbol{a}$	Vector of parameters of local function

λ_i	Degree of matching in the i^{th} rule
f_i	Local function in the i^{th} rule
d_n	Distance of n^{th} data point toward the origin of coordinate system
$\boldsymbol{pop}_i$	i^{th} individual in the population of potential solutions
$\boldsymbol{pos}_i(t)$	Position of particle i at time t
$\boldsymbol{vel}_i(t)$	Velocity of particle i at time t
$\boldsymbol{p}_{best_i}(t)$	Personal best solution found by particle *i at time t*
$\boldsymbol{g}_{best_i}(t)$	Global best solution *at time t*
Q	Performance index of model
n_i	Number of variables in input space of i^{th} rule
$\boldsymbol{x}_{reduced_i}$	Reduced input space in the antecedent of i^{th} rule
$\boldsymbol{v}_{reduced_i}$	Corresponding prototype of the i^{th} rule in the reduced space
β	Reduction parameter
W	Connections' weight matrix
t	t-norm
s	t-conorm
c_1	Number of clusters defined in input space
c_2	Number of clusters defined in output space
CR	Crossover rate
$\boldsymbol{a}_k$	Vector of fuzzy sets formed from input $\boldsymbol{x}_k$

$\boldsymbol{b}_k$	Output fuzzy sets generated by logic processing core (GP)
$\boldsymbol{t}_k$	Vector of fuzzy sets formed from output $target_k$

Chapter 1 Introduction

The primary purpose of fuzzy modeling is to achieve a set of local input-output relations that describe a process. The most commonly used fuzzy models are fuzzy rule-based systems (FRBS), consisting of a collection of fuzzy rules with a certain model structure. There are two main approaches for generating fuzzy rules, which rely on: (i) knowledge acquisition from human experts and (ii) knowledge discovery from data (data-driven). In the first approach, we can perform fuzzy modeling by extracting knowledge from human experts and transforming the expertise into membership functions and rules. The resulting system can then be tuned by monitoring its performance through trial and error. However, the dependency of the system on human experience results in some issues. First, even when human experts exist, their knowledge is often incomplete and episodic rather than systematic. Second, there is no formal and effective way of knowledge acquisition. For example, determining the proper number of rules and partitioning the input feature space remain challenging. Third, we want a system to have the learning ability to update and fine-tune itself based on newly arriving information. Therefore, in the second approach, data-driven FRBS, the researchers have been trying to automate the modeling process based on numerical training data.

1.1. Motivation

Among the variations of FRBS, Takagi–Sugeno–Kang (TSK) fuzzy systems [1] provide a helpful framework to model by decomposition of a nonlinear system into a collection of local linear models. Traditional mathematical models may fail to describe many complex nonlinear

systems' behaviors, while the potential applications of TSK fuzzy models for complex systems are vast. However, establishing the TSK fuzzy systems on the basis of data has its own concerns. Identifying the model is one of the critical issues in fuzzy modeling, which requires two main stages: (i) structural identification and (ii) parametric optimization. Structural identification is mainly concerned with the number of rules, selection of variables forming the antecedents of the rules, proper partition of the input feature space, and determination of membership functions for each variable in the antecedent parts. On the other hand, parametric optimization deals with the adjustment of system parameters, such as the coefficients of local polynomial functions in the consequences of rules, the modal values of each membership function in the antecedents. From another point of view, identifying a fuzzy model can be formulated as a search problem in multidimensional space, where each point represents a potential fuzzy model with different structures and the corresponding parameters. Due to the capability of searching multidimensional solutions, evolutionary algorithms (EAs) [2], such as genetic algorithms (GAs) [3], differential evolution (DE) [4], and genetic programming (GP) [5], have been utilized in evolutionary fuzzy modeling.

One of the motivations for structural and parametric optimization of TSK fuzzy models in this thesis is to obtain a TSK model of less complexity which retains accuracy. One alternative to achieve this aim is structuralization of fuzzy rules on both the antecedents and consequences through the efficient arrangement of input spaces together with the allocation of orders of polynomials [64]. Two modeling resources are employed, namely, (i) the total order of polynomials and (ii) the fraction of the overall number of input variables in the original space. Under the given modeling resources, this methodology involves the design of a TSK model, focusing on selection of subsets of input variables in antecedents and assignment of orders of

polynomials in consequences of the rules, in such a way that the performance index of the root mean squared error (RMSE) becomes minimized. On the one hand, structural arrangements (structural optimization) of the input spaces, over which the antecedents of rules are formed, reflect the nature of data located in the original input space. In other words, structural optimization of antecedents leads to the retention of the most influential input variables across different rules. Arranging the orders of polynomials in the consequences is also nicely reflective of the characteristics of the local input-output relationships. The structuralization helps to better express important input variables as well as less complex (more readable) local functions because they consist of low orders, say, 0, 1, 2, or 3, instead of higher orders. Unlike the literature, the arrangement of input spaces may result in variability of dimensionality of input space in different rules. On the other hand, this process is driven by the RMSE accuracy criterion that is treated as a fitness function. Therefore, both accuracy and complexity of the model is improved.

The functional fuzzy models (TSK) are utilized to describe complex non-linear systems by a collection of rules whose consequences are local functions. These functions could be linear, non-linear, differential equations, or neural networks [6]. Although the model with linear consequences possesses the universal approximation property, but in practice, the number of rules to achieve the desired accuracy is high. In the current literature, most of the TSK models with polynomial (linear) consequences have been studied; however, the TSK models' design with non-linear consequences has not been discussed notably. Another motivation in this thesis is concerned with the development of data-driven FRBS by involving a family of non-linear and linear local models rather than only linear models. The main advantage of using non-linear consequences is its capability to approximate highly non-linear functions using a small number of fuzzy rules. In other

words, the proposed modification can reduce the model's complexity preserving the desired accuracy.

Applying TSK fuzzy models to real-world problems raises more challenges. While traditional machine learning datasets have been low in feature dimensionality, many high-dimensional datasets have been used due to data analysis requirements and information technology. Examples of such application domains include geographic information systems (GIS), text documents, DNA microarray data, and image processing. High-dimensional data exhibit characteristics that differ from low-dimensional data, and these characteristics sometimes seem counterintuitive. One of the leading causes of high-dimensional data's differing properties is the phenomenon of distance concentration [7-8]; that is, all pairwise distances between the data points tend to be very similar or identical in higher dimensions. Thus, any algorithm that uses a distance function is strongly affected by this issue; e.g., membership degrees of observations be proportionally distributed across all the clusters in the Fuzzy C-Means (FCM) clustering algorithm. As the development of TSK models is fundamentally based on the FCM algorithm, this phenomenon negatively impacts the model's performance. Motivated by this idea, we want to form more distinct and meaningful clusters in the TSK models' design to cope with the issue of distance concentration. This is achieved by utilizing FCM algorithm independently to each variable rather than to all variables at once [65].

1.2. Objectives and Originality

The main objectives of this dissertation are outlined as follows:

- Developing a data-driven fuzzy rule-based modeling via efficient arrangement (reduction) of the input spaces over which the antecedents of rules are formed and allocating the orders of local polynomial functions across the consequences of the rules.
- Stressing complexity management and maintaining the approximation abilities of the FRBS using structural arrangements of antecedents and consequences across the rules.
- Investigating the impact of structuralizing the antecedents or the consequences of rules on the model performance and deciding which one is more efficient.
- Introducing a design procedure realizing logic processing through genetic programming in the form of both structural and parametric optimization.
- Development of data-driven fuzzy models based on high-dimensional data.
- Addressing the structural and parametric optimization of fuzzy rules with non-linear consequences, rather than only linear consequences.

The originality of this dissertation can be identified as follows:

- Bring design flexibility to the FRBS by admitting variability of input spaces in the antecedents of different rules and variability of orders in the local functions forming the consequences.
- The rules are refined/revised based on the newly introduced modeling resources, namely, the total order of polynomials over all the rules and the fraction of overall number of input variables in the original space.
- Bring flexibility to the logic-oriented structure of fuzzy models involving logic operators such as Xor, Nor, Nand, Equivalence, and Implication.
- A variety of *t*-norms and *t*-conorms are employed in the evolutionary process to form the optimal model.

- Although it is essential, high-dimensional data has not yet been discussed enough in the literature of TSK models. Independent clustering of single variables rather than all variables at once, and then their combination using GP is a new methodology for the formation of useful fuzzy models.
- Linear functions in the consequences of fuzzy rules are replaced with affine non-linear consequences to generalize the traditional TSK FRBS. This is performed based on allocating some non-linear functions such as sine, cosine, exponential, and square root across various fuzzy rules as their consequences.
- In contrast to traditional TSK models, which have the same type of functions in all the consequences of rules, there is a variability of local models standing in the model's consequences proposed here.
- The main advantage of using non-linear consequences is its capability to approximate highly non-linear functions using a small number of fuzzy rules. In other words, the proposed modification can reduce the model's complexity preserving the desired accuracy.

1.3. Dissertation organization

This dissertation is structured into the following chapters:

Chapter 2 briefly reviews some helpful background involved in this thesis, including fuzzy clustering, fuzzy modeling, distance concentration issue in high-dimensional data, and population-based optimization methods.

Chapter 3 covers the development of different TSK models through PSO. This framework stresses the need for and benefits of structural refinement of fuzzy rules to make the model less complex while still retaining its accuracy. Two different ways of structuralizing the antecedents

and consequences of the rules, based on the newly introduced modeling resources, are proposed: (i) the arrangement of input spaces in the antecedents of the rules and (ii) the arrangement of the orders of polynomials in the consequences of the rules.

Chapter 4 proposes a design procedure based on genetic programming realizing logic processing in the form of structural and parametric optimization. This chapter also addresses the distance concentration issue in the design of high-dimensional fuzzy models, where the increase of dimensionality may converge all the pairwise distances (dissimilarities) to the same value, leading to failure of the fuzzy model.

Chapter 5 introduces an identification framework to generalize Takagi–Sugeno–Kang (TSK) fuzzy rule-based systems (FRBS). This framework stresses the need to use non-linear functions (rather than linear) standing in the fuzzy rules' consequences. The proposed modification reduces the model's complexity (number of rules) while preserving the desired accuracy.

Finally, the work presented in this dissertation is summarized in chapter 6, and we identify some promising directions for future research.

Chapter 2 Preliminaries

This chapter briefly covers some essential prerequisites that make this thesis self-contained and easy to follow. In Section 2.1, we elaborate on fuzzy clustering, in particular the FCM algorithm. Then, in Section 2.2, we cover fundamental aspects of fuzzy modeling, particularly fuzzy rule-based models. Next, the distance concentration issue in high-dimensional data is elaborated in Section 2.3. Finally, in Section 2.4, we detail the population-based optimization techniques utilized in our research.

2.1 Fuzzy clustering

Data clustering is the task of grouping a set of data in such a way that data in the same group, so-called cluster, are more similar (in some sense) to each other than to those in other clusters. In granular computing, clustering is one of the most widely used methods of forming information granules. Depending upon the nature of the underlying clustering method, the information granules produced arise as sets, fuzzy sets, or rough sets. The FCM [9] is one of the mostly-used clustering algorithms which forms a collection of fuzzy sets. Let us discuss the formulation of the FCM and its development in a nutshell. The FCM algorithm starts with a desired number of clusters and random assignment of the data points to the clusters. Thus, all the data points have a membership value for each cluster. By iteratively updating prototypes (i.e., cluster centers) and membership values, the algorithm aims to guide the prototypes to the optimal locations in the data space. Suppose we have the data set as $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)^{\mathrm{T}}$, $\boldsymbol{x}_k$ is the k^{th} data point in the n-dimensional

space $\boldsymbol{R}^n$. The generic version of the FCM algorithm minimizes the following objective function Q with the weighted Euclidean distance as

$$Q = \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^{m} \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \tag{2.1}$$

with the distance is expressed as

$$\|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 = \sum_{j=1}^{n} \frac{(x_{kj} - v_{ij})^2}{\sigma_j^2} \tag{2.2}$$

where σ_j is a standard deviation of the j^{th} variable of the data, and fuzzification coefficient m is usually greater than 1. Obviously, the distance function's choice impacts the geometry of the clusters, which entails modeling capabilities supported by the ensuing rule-based models. The data is partitioned into c clusters come in the form of the partition matrix $U = [u_{ik}]_{c \times N}$, $i = 1, 2, \ldots, c$; k=1, 2,.., N, and a collection of prototypes represented as V=($\boldsymbol{v}_1$, $\boldsymbol{v}_2$, …, $\boldsymbol{v}_c$)$^{\text{T}}$. The k^{th} data item is described in terms of the k^{th} column membership grades in the partition matrix. The algorithm proceeds iteratively by first updating the cluster prototypes:

$$v_{it} = \frac{\sum_{k=1}^{N} u_{ik}^{m} x_{kt}}{\sum_{k=1}^{N} u_{ik}^{m}} \tag{2.3}$$

where t = 1, 2,…, n, and then followed by the re-computation of the partition matrix with the new prototypes using the following membership function:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\boldsymbol{x}_k - \boldsymbol{v}_i\|}{\|\boldsymbol{x}_k - \boldsymbol{v}_j\|} \right)^{\frac{2}{(m-1)}}} \tag{2.4}$$

The algorithm is terminated once the given termination criterion is met, commonly when the change in the partition matrix is less than some specified threshold value.

When the data are in the form of $(\boldsymbol{x}_k, y_k)$, k=1, 2, …, N, where $\boldsymbol{x}_k$ is the k^{th} data point in the n-dimensional input space $\boldsymbol{R}^n$ and y_k is its corresponding single output, they are combined into (n+1)-dimensional vectors $\boldsymbol{p}_k$=[$\boldsymbol{x}_k$, y_k]. Then the FCM algorithm is completed in n+1 dimensional space and produces the prototypes expressed in the form of $(\boldsymbol{v}_i, w_i) \in \boldsymbol{R}^{n+1}$, i = 1, 2,…, c, where $\boldsymbol{v}_i$ and w_i are the prototypes positioned in the input and output spaces, respectively [6].

2.2 Fuzzy modeling

In general, fuzzy models operate at a level of information granules, i.e., fuzzy sets. In this way, they constitute highly abstract and flexible constructs that provide many modeling advantages compared to the traditional computing techniques that deal directly with numerical information. The combination of approximation and interpretability capabilities of fuzzy models makes them powerful machine learning tools, particularly for research areas where the need to understand the reasoning behind model behavior is crucial. Fuzzy models are inherently able to represent and process uncertainty with this processing of information granules, which are apparent in all parts of the real world.

We focus on the fuzzy model's general architecture in this section to elaborate on its functional elements. A general fuzzy model has two essential functional parts: (i) input and output interfaces and (ii) a processing core. The interfaces enable communication between the fuzzy model's conceptual structure and the physical world of measured variables. The input interface realizes perception, where input variables are transformed into an internal format of information granules (fuzzy sets) understood by the logic-processing core. The output interface communicates the processing results in a form understood by the external world (modeling environment). These

actions can be referred to as fuzzy encoding (input interface) and decoding (output interface) or, more traditionally, fuzzification and defuzzification. The logic-processing core forms the most crucial component of the fuzzy model, composed of a knowledge base containing the structure and details of system behavior, realizing inference through granular computation. The overall model is shown in Figure 2.1.

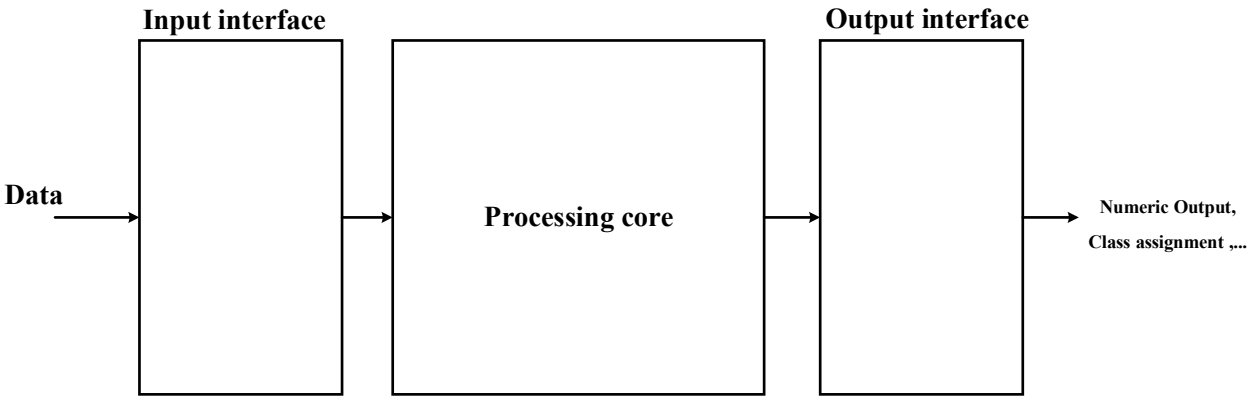


Figure 2. 1. A general view at the underlying architecture of fuzzy models [6]

Rule-based models play a central role in fuzzy modeling. These models are mainly used to describe and handle complex nonlinear relationships by formulating "if-then" rules that are overlapped through input and output space and contain extractions of knowledge in the following form:

If antecedent proposition then consequent proposition (2.5)

In terms of taxonomy, fuzzy rule-based models can be roughly divided into several families, depending on the type of fuzzy sets used in the rules. The two most widely-used architectures are Linguistic (Mamdani) [10] and Functional Takagi–Sugeno–Kang (TSK) fuzzy models [1]. The main difference between these two is in the form of the consequent part of rules. The Mamdani

model employs linguistic terms (fuzzy sets), whereas the TSK model uses the polynomial local function of antecedent variables in the consequent part.

2.2.1 Linguistic (Mamdani) fuzzy models

The following expression is the common form of linguistic fuzzy model:

$$R_i : \text{if } X \text{ is } A_i \text{ and } Y \text{ is } B_i \text{ then } Z \text{ is } C_i \tag{2.6}$$

where R_i denotes the i^{th} rule, i=1, 2,…, c, and c is the total number of rules. X, Y, and Z are linguistic variables with base variables x, y, and z. A_i, B_i, and C_i are fuzzy sets on X, Y, *and* Z, respectively. The aggregation of the rules is realized as a union of the Cartesian products of the fuzzy sets standing in the antecedents and consequents parts of the individual rule. There are several options to aggregate the rules, but often rule aggregation is carried out using minimum or product t-norms. Here, "if-then" rules are defined as Cartesian products using the minimum or product t-norms and the maximum t-conorms to perform aggregation rules. In what follows, we illustrate one of the essential linguistic models is called the min-max models.

Figure 2.2 illustrates the main processing steps of the min-max linguistic model. As this figure displays, the principal stages of the model are as follows: (i) antecedent matching: for each rule R_i, i=1,2,….c, compute the degree of matching by using possibility measure, i.e. m_i = max [min(A (x), $A_i(x)$)] and n_i = max [min($B(x)$, $B_i(x)$)], (ii) antecedent aggregation: for each rule R_i, compute the rule activation degree by conjunctively or disjunctively operating on the corresponding degrees of matching: $\lambda_i = \min(m_i , n_i)$, (iii) rule result derivation: for each rule R_i, , compute the corresponding inferred value based on its antecedent aggregation and the rules semantics chosen. $C_i' = \min(\lambda_i , C_i)$, and (iv) rule aggregation: compute the inferred value from the complete set of rules by

aggregating the result of the inferred values derived from individual rules, $C(y) = \max_{i=1,2,\ldots,c} (C_i')$. In case a numeric outcome of inference is required, a certain decoding is completed. Despite the evident simplicity of the overall construct outlined above, it supports efficient nonlinear input-output mapping.

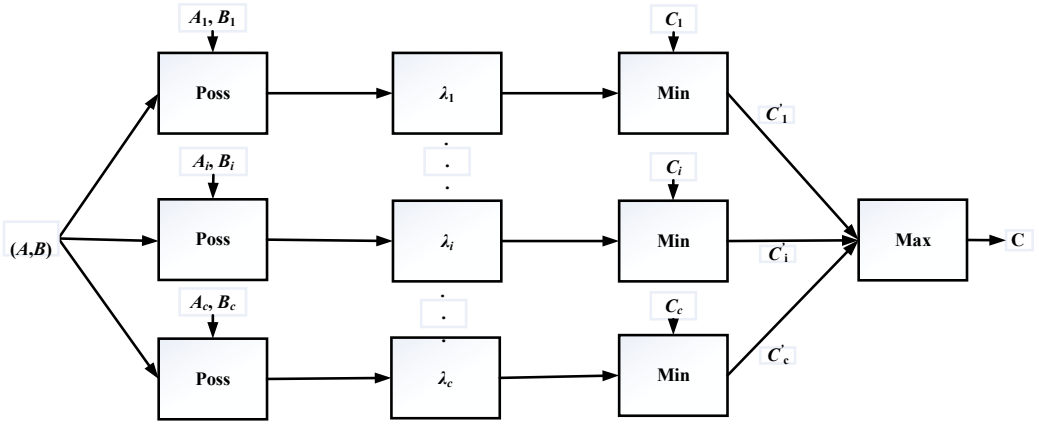


Figure 2. 2. Min-max fuzzy linguistic model [6]

2.2.2 Functional TSK fuzzy models

The most established and well-studied form of FRBS is the TSK model. The TSK fuzzy model provides a systematic approach for generating fuzzy rules from a given input-output data set. The following expression is the common form of TSK style rules:

$$R_i : \text{if } x_1 \text{ is } A_{i,1} \text{ and } \ldots x_n \text{ is } A_{i,n} \text{ then } y_i = f_i(\boldsymbol{x}, \boldsymbol{a}_i) \tag{2.7}$$

where $i = 1, 2,\ldots, c$, and c is the number of rules, $\boldsymbol{x}$ is a n-dimensional input variable. A_i is the membership function of i^{th} fuzzy set in the input space and $\boldsymbol{a}_i$ is a vector of coefficients in the local function $f_i(\boldsymbol{x}, \boldsymbol{a}_i)$. As earlier mentioned, the membership functions can be estimated relying on expert experience or by admitting a data-driven approach. In the second alternative, the clustering algorithm plays an essential role in generating clusters (prototypes) and determining the model's

rule structure. In the consequent part, y_i is the output of the i^{th} rule described by some local function $f_i(\boldsymbol{x}, \boldsymbol{a}_i)$, which is typically regarded as a linear function or simply as some constant values.

When arranging all the rules together involving their antecedent parts, the output of the model, say $\hat{y}$, is aggregated by taking the weighted average of the output of each rule as follows,

$$\hat{y}=\frac{\sum_{i=1}^{c} A_i(\boldsymbol{x}) f_i(\boldsymbol{x}, \boldsymbol{a}_i)}{\sum_{i=1}^{c} A_i(\boldsymbol{x})} \tag{2.8}$$

If we consider utilizing FCM clustering to form the membership functions, it holds that $\sum_{i=1}^{c} A_i(\boldsymbol{x})=1$. The output of the fuzzy models can be rewritten in the following expression,

$$\hat{y}=\sum_{i=1}^{c} A_i(\boldsymbol{x}) f_i(\boldsymbol{x}, \boldsymbol{a}_i) \tag{2.9}$$

Some orders (types) of the polynomials are considered:

$$\text{Type 0 (constant): } f_i(\boldsymbol{x},\boldsymbol{a}_i)=a_{i0} \tag{2.10}$$

$$\text{Type 1 (linear): } f_i(\boldsymbol{x},\boldsymbol{a}_i)=a_{i0} + \boldsymbol{a}_{i1}^{\mathrm{T}}\boldsymbol{x} \tag{2.11}$$

$$\text{Type 2 (quadratic): } f_i(\boldsymbol{x},\boldsymbol{a}_i)=a_{i0} + \boldsymbol{a}_{i1}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{a}_{i2}^{\mathrm{T}}\boldsymbol{x}^2 \tag{2.12}$$

$$\text{Type 3 (cubic): } f_i(\boldsymbol{x},\boldsymbol{a}_i)=a_{i0} + \boldsymbol{a}_{i1}^{T}\boldsymbol{x} + \boldsymbol{a}_{i2}^{T}\boldsymbol{x}^2 + \boldsymbol{a}_{i3}^{T}\boldsymbol{x}^3, \tag{2.13}$$

where $\boldsymbol{a}_{ij}$ is a vector of coefficients in j^{th} term of rule i, and $\boldsymbol{x}^k=\boldsymbol{x}\odot\boldsymbol{x}\ \ldots\odot\boldsymbol{x}=[x_1^k, x_2^k, \ldots, x_n^k]^{\mathrm{T}}$, k=2, 3,... The symbol $\odot$ denotes an element-wise multiplication operator of two vectors. Take, for example, vector $\boldsymbol{x}=[x_1, x_2, x_3]^{\mathrm{T}}$, so $\boldsymbol{x}^2=\boldsymbol{x}\odot\boldsymbol{x}=[x_1^2, x_2^2, x_3^2]^{\mathrm{T}}$ and $\boldsymbol{x}^3=\boldsymbol{x}\odot\boldsymbol{x}\odot\boldsymbol{x}=[x_1^3, x_2^3, x_3^3]^{\mathrm{T}}$.

In the original TSK fuzzy model, $f_i(\boldsymbol{x},\boldsymbol{a}_i)$ in the consequent parts of the rules is adopted as a linear function as follows,

$$f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0}+a_{i1}x_{i1}+a_{i2}x_2+\ldots+a_{in}x_n \tag{2.14}$$

The parameters of the local function are identified by minimizing the Least Square Error (LSE) optimization criterion [9]. In some cases, for the sake of simplification, a_{i1}, a_{i2},…, and a_{in} can be defined as zeros so that the function is simplified as a constant value.

Let us see the process of determining parameters in detail. The parameters of a linear local function can be shown in the form of $\boldsymbol{a}_i = [a_{i0}, a_{i1},\ldots, a_{in}]$. The output of the model is expressed in the following form:

$$\hat{y}=\sum_{i=1}^{c} A_i(\boldsymbol{x})\, .\boldsymbol{a}_i^{\mathrm{T}} \begin{bmatrix}1\\ \boldsymbol{x}\end{bmatrix} = \sum_{i=1}^{c} \boldsymbol{a}_i^{\mathrm{T}} \begin{bmatrix}A_i(\boldsymbol{x})\\ A_i(\boldsymbol{x})\boldsymbol{x}\end{bmatrix} \tag{2.15}$$

Suppose

$$z_i(\boldsymbol{x})=\begin{bmatrix}A_i(\boldsymbol{x})\\ A_i(\boldsymbol{x})\, x_1\\ A_i(\boldsymbol{x})\, x_2\\ \cdots\\ A_i(\boldsymbol{x})\, x_n\end{bmatrix} \tag{2.16}$$

so

$$\hat{y}=\sum_{i=1}^{c} \boldsymbol{a_i}^{T} z_i(\boldsymbol{x})=\sum_{i=1}^{c} z_i^{\mathrm{T}}(\boldsymbol{x})\, \boldsymbol{a}_i \tag{2.17}$$

Let us use the following vector notation to collect all parameters of the models

$$\boldsymbol{a} = \begin{bmatrix}\boldsymbol{a}_1\\ \boldsymbol{a}_2\\ \boldsymbol{a}_3\\ \cdots\\ \boldsymbol{a}_c\end{bmatrix} \tag{2.18}$$

and

$$f(\boldsymbol{x}) = \begin{bmatrix} z_1(\boldsymbol{x}) \\ z_2(\boldsymbol{x}) \\ z_3(\boldsymbol{x}) \\ \cdots \\ z_c(\boldsymbol{x}) \end{bmatrix} \tag{2.19}$$

The collection of N-input-target data is then organized in the following matrix format:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdots \\ y_N \end{bmatrix} \tag{2.20}$$

and

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}_1) \\ \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}_2) \\ \cdots \\ \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} z_1{}^{\mathrm{T}}(\boldsymbol{x}_1) & z_2{}^{\mathrm{T}}(\boldsymbol{x}_1) & \cdots & z_c{}^{\mathrm{T}}(\boldsymbol{x}_1) \\ z_1{}^{\mathrm{T}}(\boldsymbol{x}_2) & z_2{}^{\mathrm{T}}(\boldsymbol{x}_2) & \cdots & z_{\mathrm{c}}{}^{\mathrm{T}}(\boldsymbol{x}_2) \\ \cdots & \cdots & \cdots & \cdots \\ z_1{}^{\mathrm{T}}(\boldsymbol{x}_N) & z_2{}^{\mathrm{T}}(\boldsymbol{x}_N) & \cdots & z_c{}^{\mathrm{T}}(\boldsymbol{x}_N) \end{bmatrix} \tag{2. 21}$$

so

$$\hat{\boldsymbol{y}} = \boldsymbol{F}\boldsymbol{a} \tag{2.22}$$

With the sum of squared error $Q = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2 = (\boldsymbol{F}\boldsymbol{a} - \boldsymbol{y})^{\mathrm{T}}(\boldsymbol{F}\boldsymbol{a} - \boldsymbol{y})$ and its minimization with respect to $\boldsymbol{a}$, the optimal estimated parameters are expressed in the following format:

$$\boldsymbol{a}_{\text{optimal}} = (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{y} \tag{2.23}$$

Another variation of the TSK model is involving the prototypes obtained from clustering to the local function. Namely, $f_i(\boldsymbol{x}, \boldsymbol{a}_i)$ stands for a local linear function interpreted as a hyperplane governed by the following expression,

$$f_i(\boldsymbol{x}, \boldsymbol{a}_i) = w_i + \boldsymbol{a}_i{}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{v}_i) \tag{2.24}$$

where $\boldsymbol{v}_i$ is a cluster (prototype) capturing the location of the rule in the input space $\boldsymbol{R}^n$ and w_i is the corresponding value in the output space, then

$$\hat{y} = \sum_{i=1}^{c} A_{\mathrm{i}}(\boldsymbol{x})\ (w_i + \boldsymbol{a}_i^{\mathrm{T}}(\boldsymbol{x}\text{-}\boldsymbol{v}_i)) \tag{2.25}$$

Let us introduce some auxiliary notation as shown below,

$$\boldsymbol{Z}_i = A_i(\boldsymbol{x})(\boldsymbol{x}\text{-}\boldsymbol{v}_i) \tag{2.26}$$

$$q = \sum_{i=1}^{c} A_{\mathrm{i}}(\boldsymbol{x}) w_{\mathrm{i}} \tag{2.27}$$

Then, the above model is concisely described in the form of

$$\hat{y} = q + \sum_{i=1}^{c} \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{Z}_i \tag{2.28}$$

We introduce the following concise notation,

$$\boldsymbol{p} = [y_1\text{-}q_1, y_2\text{-}q_2, \ldots, y_N\text{-}q_N]^{\mathrm{T}} \tag{2.29}$$

In the sequel, the parameters of the model are arranged into the cn-dimensional vector.

$$\boldsymbol{a} = [a_{11}, a_{12,\ldots,} a_{1n,} a_{21,} a_{22\ldots,} a_{2n,,\ldots,} a_{c1,} a_{c2,\ldots} a_{cn}]^{\mathrm{T}} \tag{2.30}$$

Furthermore, the data are structured in the matrix format

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}_{11} & \boldsymbol{z}_{12} & \cdots & \boldsymbol{z}_{1c} \\ \boldsymbol{z}_{21} & \boldsymbol{z}_{22} & & \boldsymbol{z}_{2c} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{z}_{N1} & \boldsymbol{z}_{N2} & \cdots & \boldsymbol{z}_{Nc} \end{bmatrix} \tag{2.31}$$

Then

$$Q = \|\mathbf{y}\text{-}\widehat{\mathbf{y}}\|^2 = \sum_{k=1}^{N} \left(\mathrm{y_k}\text{-}q_k\text{-}\sum_{i=1}^{c} \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{z}_{ki}\right)^2 = (\boldsymbol{p}\text{-}\boldsymbol{Z}\boldsymbol{a})^{\mathrm{T}}(\boldsymbol{p}\text{-}\boldsymbol{Z}\boldsymbol{a}) \tag{2.32}$$

Minimizing (32), we estimate the parameters $\boldsymbol{a}$ by minimizing the LSE as,

$$\boldsymbol{a}_{\text{optimal}} = (\boldsymbol{Z}^{\text{T}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\text{T}}\boldsymbol{p} \tag{2.33}$$

2.2.3 Other forms of fuzzy models

Depending upon the nature of the problem and properties of the domain knowledge, rules may come in the different formats, such as certainty-qualified and gradual rules [6]. In the certainty-qualified rules, instead of allocating full confidence in the rules' validity, we allow to treat them as being satisfied (valid) at some level of confidence. The degree of uncertainty leads to certainty-qualified expressions of the following form:

if X is A and Y is B then Z is C with certainty μ (2.34)

where $\mu \in [0,1]$ denotes the degree of certainty of this rule. If μ =1, we say that the rule is certain.

Rules may also involve gradual relationships between objects, properties, or concepts. For example, the rule

The more X is A; the more Y is B (2.35)

expresses a relationship between changes in Y triggered by the changes in X. In these rules, rather than expressing some association between antecedents and consequences, we capture the tendency between the information granules; hence the term of graduality occurring in the antecedent and consequent part. For instance, the graduality occurs in the rule "the higher the income, the higher the taxes'' or typically, "the higher the horsepower, the higher the fuel".

2.3 Distance concentration issue in high-dimensional data

High-dimensional data analysis gives rise to many challenges. One such that has come to gain much attention recently is the distance concentration phenomenon, which is the inability of distance functions to distinguish data points well in high dimensions. That is, as the data dimensionality increases, all the pairwise distances (dissimilarities) may converge to the same value [7]. The lack of contrast between the nearest and the furthest points affects each machine learning area where high-dimensional data processing is required, e.g., data clustering. It can be formally described in the following way: Let X_n be an n-dimensional random vector with i.i.d. (independent and identically distributed) components of any distribution, and let $d_n(\boldsymbol{x})$ denote the distance of $\boldsymbol{x}\in\boldsymbol{R}^n$ to the origin of the coordinate system with a suitable distance measure; for instance, Euclidean distance. Let $d_n^{(max)}$ and $d_n^{(\min)}$ denote the largest and smallest distance of a point in a sample toward the origin of the coordinate system. Then, the following holds,

$$\lim_{n\to\infty} \operatorname{var}\left(\frac{d_n(X_n)}{\mathrm{E}(d_n(X_n))}\right) = 0 \;\Rightarrow\; \frac{(d_n^{(\max)} - d_n^{(\min)})}{d_n^{(\min)}} \to_p 0 \tag{2.36}$$

where $\to_p$ denotes convergence in probability. In other words, when the relative variance (i.e., relative with respect to the mean distance) converges to zero with increasing dimensionality of the variables, then the relative difference of the closest and farthest point in the data goes to zero, where the number of variables increases [8]. Consequently, clustering algorithms based on the spatial structure of the data (quantified in some distance) cannot work well if all data tend to be equally distant.

2.4 Population-based optimization

There are several problems, particularly in the area of engineering, whose objective functions are complex and non-differentiable. Although there are some analytical and numerical optimization methods, there are no universal approaches. For FRBS, using gradient-based methods is not of much use, primarily due to their non-linear nature. This deficiency implies using population-based optimization approaches, such as evolutionary algorithms (EAs) or swarm intelligence. Despite gradient-based algorithms, population-based approaches can escape from the local optimum due to their exploration characteristic. As these methods have both exploration and exploitation potentials, they are suitable methods of constructing fuzzy models. In light of this, some of those approaches, such as genetic algorithm (GA), Genetic programming (GP), differential evolution (DE), and particle swarm optimization (PSO), are introduced in this section.

2.4.1 Genetic algorithm (GA)

GA is a population-based technique of optimization which belongs to the class of evolutionary algorithms [11]. GA explores the search space by using the mechanisms encountered in natural evolution: mutation, crossover (recombination), and selection. In nature, individuals' fitness depends on their phenotype, which is directly influenced by their genotype (chromosome). Individuals with greater fitness have a greater chance of survival and also a more comprehensive range of mating partners to choose from within the population. The new individuals are generated by genetic operators, namely crossover and mutation. The space of feasible solutions is called search space. Each point in the search space represents one feasible solution. The coding of solutions is problem-dependent, say real, integer, and binary encoding. The implementation of GA

requires determining some fundamental issues, say, chromosome representation, selection function, genetic operators, and fitness function. The basic structure of GA includes three main steps: (i) forming the initial population, (ii) evaluating the fitness of the individuals, and (iii) the construction of an intermediate population through the selection and recombination process. The chromosome representation of the population is the primary concern in GA. The common representation uses fixed-length and binary coded strings. However, for more complex problems, using a real-coded string is favorable as the representation is more natural to domains' nature. Some of the most well-known selection methods are proportional selection, tournament selection, and ranking selection. After the selection process has been carried out to determine the parent chromosomes, the intermediate population's construction is handled over the recombination process, which involves two operators called crossover and mutation. The crossover operator plays a vital role in GA, where this operator is involved in the sharing of information between the chromosomes (i.e., exploitation); it combines the features of parents to form offspring, with the possibility that the good parent may generate better chromosome. Next, the mutation operators' role is to prevent GA's premature convergence to local solutions (i.e., exploration). The mutation operator restores lost or unexplored genetic material into the population. These two operators are not usually applied to all pairs/single chromosomes in the immediate population. A random choice is made according to the probability defined by the mutation rate (e.g., 0.1) and crossover rate (e.g., 0.85). Finally, another selection technique called the elitist strategy is usually implemented after crossover and mutation. By using the elitist strategy, we assure that the best performing chromosome always survives intact from one generation to the next.

2.4.2 Genetic programming (GP)

GP [12] is a population-based method that optimizes a problem using the iterative improvement of candidate solutions. It employs a population of non-fixed length possible solutions, so-called programs, to the given problem. These programs can be expressed as parse trees rather than strings. GP uses a set of predefined functions and terminals to construct individuals. For example, the function set can contain the basic arithmetic operations (+,-,*, /, etc.), Boolean logic functions (and, or, not, etc.), or any other mathematical functions. The terminal set contains the arguments for the functions and can consist of numerical constants, logical constants, variables, etc. Generally, a GP algorithm follows the coming process. An initial population of GP individuals is developed randomly, and a predefined fitness function is employed to evaluate their individuals' finesses. The fittest individuals of the current population are selected then as the parents using a selection method. Next, genetic operators (crossover and mutation) are applied to the parents to generate offspring. Following that, the current population is replaced by a population of offspring. This process proceeds iteratively until a termination condition has been met and the individual with the best fitness returned. Koza [12] suggested three methods for initial population generation: the full, grow and ramped half-and-half methods. The initial individuals in both full and grow methods are formed not to exceed a user-specified maximum depth. The full method forms full trees (i.e., all the leaves are at the same depth) in which nodes are randomly taken from the function set until the full tree set is reached. Beyond that depth, only terminals can be chosen, whereas in the grow method, nodes can be taken from both function and terminal sets until the depth limit is met, then the leaves can be chosen from terminals. Because neither grow or full method provides a vast array of sizes or shapes on their own, a combination called ramped half-and-half was proposed. Half the initial population is constructed using the full, and half is constructed using the

grow method. This is done using a range of depth limits (hence the term "ramped") to help ensure that the trees have various sizes and shapes.

2.4.3 Differential evolution (DE)

Differential Evolution (DE) [11] is another popular EA strategy with a clear resemblance with GA and GP, but it differs considerably in the sense that distance and direction information from the current population is issued to guide the search process. Another difference is that mutation is applied first to generate a trial vector, and then a crossover operator is utilized to produce offspring. In DE, the population evolves towards an optimum solution through a series of evolution operations, such as mutation, crossover, and selection; that is, DE is looking for some solutions, which are the best. In the beginning, the population ***pop*** is initialized randomly. Then, in each generation, a mutant vector is generated for the i^{th} individual in ***pop*** denoted as

$$\boldsymbol{mv}_i = \boldsymbol{pop}_{q_1} + F.\,(\boldsymbol{pop}_{q_2} - \boldsymbol{pop}_{q_3}) \tag{2.37}$$

$$\boldsymbol{mv}_i = \boldsymbol{r}_{best} + F.\,(\boldsymbol{pop}_{q_1} - \boldsymbol{pop}_{q_2}) \tag{2.38}$$

where q_1, q_2, and q_3 are random integer indexes for the individuals in the population, and F is the differential weight positioned in the interval [0, 2] and treated as control parameters. $\boldsymbol{r}_{best}$ is the best individual of the current generation. Formulas (2.37) and (2. 38) are two common mutation strategies used in DE.

The crossover operation is used to increase the diversity of the produced mutation vector, which tries to adjust the j^{th} variable in the i^{th} mutation vector as

$$\boldsymbol{p}_{ij}=\begin{cases}\boldsymbol{mv}_{ij} & , rand(0,1)\le CR \text{ or } j=j_{rand}\\ \boldsymbol{r}_{ij} & , \text{otherwise.}\end{cases} \tag{2.39}$$

where $rand(0,1)$ stands for a random number coming from the uniform distribution within the range [0, 1] and j_{rand} is an integer index that is selected randomly from the uniform distribution over the range [1, n], where n is the number of features. CR is the crossover rate, which is randomly generated between 0 and 1. Individual $\boldsymbol{r}$ is either $\boldsymbol{pop}_{q1}$ or $\boldsymbol{r}_{best}$, depending on the used mutation strategy. Once the crossover operation has been done, a selection process determines the survivors for the next generation. In that process, better individuals (evaluated by a fitness function Q, and minimization of Q is pursued) obtained by mutation and crossover operations are selected as

$$\boldsymbol{pop}_i=\begin{cases}\boldsymbol{p}_i & , Q(\boldsymbol{p}_i)<Q(\boldsymbol{pop}_i)\\ \boldsymbol{pop}_i & , \text{otherwise.}\end{cases} \tag{2.40}$$

2.4.4 Particle swarm optimization (PSO)

The PSO [11] is a well-known strategy in swarm computation that optimizes a problem using the iterative improvement of candidate solutions, so-called particles. Cooperation is one of the essential characteristics of this algorithm; that is, each particle of the population moves in search space in the current direction influenced by two factors: (i) the position of the best particle in the swarm and (ii) the best position that is experienced so far by the particle itself. Cooperation helps in moving the particle towards the best positions. During this process, the position and velocity of the particle are updated as follows:

$$\boldsymbol{pos}_i(t+1)=\boldsymbol{pos}_i(t)+\boldsymbol{vel}_i(t+1) \tag{2.41}$$

$$\boldsymbol{vel}_i(t+1)=w\,\boldsymbol{vel}_i(t)+c_1\,\boldsymbol{r_1}\,.\left(\boldsymbol{p}_{best_i}(t)-\boldsymbol{pos}_i(t)\right)+c_2\,\boldsymbol{r_2}\,.\left(\boldsymbol{g}_{best_i}(t)-\boldsymbol{pos}_i(t)\right) \tag{2.42}$$

The index of the particle is represented by i, and the position and velocity of particle i at the time t are respectively described by the vectors $\boldsymbol{pos}_i(t)$ and $\boldsymbol{vel}_i(t)$. The parameters w, c_1, and c_2 ($0 \le w \le 1.2$, $0 \le c_1 \le 2$, and $0 \le c_2 \le 2$) are user-defined coefficients. The values of r_1 and r_2 are vectors of random numbers between 0 and 1, generated for each particle. $\boldsymbol{p}_{best_i}(t)$ and $\boldsymbol{g}_{best_i}(t)$ are the personal best solution found so far by the particle i, and the global best solution at the time t, respectively. The symbol "$\cdot$" denotes an element-wise multiplication operator of two vectors.

2.5 Conclusions

This chapter covers the fundamental concepts and algorithms essential in designing fuzzy models introduced in this dissertation. The fuzzy clustering algorithm (particularly FCM) is used as the foundation to form fuzzy rule-based models and was employed in all the architectures introduced in this thesis. This chapter has also elaborated in detail the fundamental aspects of fuzzy modeling, particularly rule-based models, and the population-based methods for solving the optimization problems. The issue of distance concentration in high-dimensional data is also discussed in this chapter. This issue leads to the failure of the standard TSK models in high dimensions.

Chapter 3 Structural Optimization of Fuzzy Rule-based Models: Towards Efficient Complexity Management

This chapter's primary aim is concerned with the structural optimization of data-driven fuzzy rule-based systems (FRBS), with the intent of their complexity management. This is accomplished in two ways: the first one involves a structuralization of the antecedents and the second one deals with a structuralization of the consequences of the fuzzy rules. More specifically, this chapter contributes to the complexity management of fuzzy models by focusing on (i) the efficient arrangement (reduction) of the input spaces over which the antecedents of rules are formed and (ii) allocating the orders of local polynomial functions across the consequences of the rules. The originality of the study comes with the flexibility of FRBS that is endowed by admitting variability of input spaces standing in the antecedents of different rules and the variability of orders of polynomials (local functions) forming the consequences of the rules.

Particle swarm optimization (PSO) is guided by the root mean squared error (RMSE) accuracy criterion to realize the efficient arrangement of input spaces and an allocation of the orders of the individual polynomials. In this optimization process, the FCM clustering algorithm is employed to create fuzzy sets in the rules' antecedents, while the LSE optimization criterion is minimized to determine the coefficients of the polynomials in the consequences. The proposed model's performance is quantified using some numeric data, including both synthetic and machine learning datasets.

3.1. Structural and parametric optimization of fuzzy rule-based systems

The design of the TSK FRBS can be regarded as an optimization task. This task is divided into two steps: (i) structural optimization and (ii) parametric optimization of fuzzy rules. Structural identification is mainly concerned with the number of rules, selection of variables forming the antecedents of the rules, and determination of membership functions for each variable in the antecedent. On the other hand, parametric identification involves estimating local polynomial functions' coefficients in the consequences and identifying each membership function's modal values in the antecedents.

Due to the ability to explore large search space, swarm and evolutionary computation have been employed to construct TSK FRBS. Setnes and Roubos [13] described a genetic algorithm (GA) to construct accurate FRBS of TSK. This method was also equipped with the rule base simplification capability to merge similar fuzzy sets and remove the redundant fuzzy sets [14]. Chen and Wong [15] proposed a binary GA to find a fuzzy system with a low number of rules and predetermined acceptable performance. However, this method could not detect input variables of little significance (relevance). To address this issue, Teng and Wang [16] employed a GA-based fuzzy system that had a dynamic structure; that is, a different number of membership functions was assigned to input variables according to their significance. If any input variables did not have extra membership functions assigned, they were discarded as dummy input variables to assure efficient fuzzy systems. Pedrycz and Song [17] presented some identification methodologies of fuzzy models based on information granulation and GA. They even determined the orders and parameters of polynomials occurring in the consequent parts in the literature [18-19]. Indeed, they incorporated the order of polynomial as a part of the optimization process. Although the fuzzy rules generated by their approaches might have various coefficients of polynomials at

consequences of different rules, the limitations were that all the rules had the same order of polynomials, and the input spaces specified by the GA were the same for all the individual rules. Other swarm and evolutionary methods such as particle swarm optimization (PSO) [20], differential evolution (DE) [21], and genetic programming (GP) [5] are also available in the literature. Tsai and and Chen [22] proposed an identification method for the TSK model by utilizing PSO. Most of the models encountered in the literature did not provide an efficient way to determine the number of fuzzy rules; however, in the mentioned work, Xie–Beni indices with a FCM clustering algorithm were adopted to find the rule number of the TSK model; then, by utilizing the PSO algorithm, the initial membership function and the consequent parameters of the fuzzy model were obtained. Tsakiridis et al. [4, 23] designed a Mamdani FRBS so-called DECO3RUM (Differential Evolution based Cooperative and Competing learning of Compact Rule-based Models) for the prediction of soil properties from soil spectral libraries. A complete overview of evolutionary methods encountered in the automatic synthesis of fuzzy systems is given in [2, 5].

A multi-objective evolutionary algorithm (MOEA) is another suitable alternative for structural optimization of FRBS which mostly focuses on the trade-off between accuracy and interpretability of model. In general, accuracy and interpretability of fuzzy systems are two contradictory objectives. Accuracy is the capability of the model to truly represent the real systems (data), whereas interpretability is the potential to express the behavior of the real model in an easily understandable way. The ideal model has both high accuracy and interpretability, but the contradictory natures of these requirements makes building an ideal model highly unlikely if not infeasible. Thus, it becomes essential to strike a sound balance between accuracy and interpretability. Ishibuchi and Nojima [24] demonstrated, using simple testing problems, which

the minimization of complexity of a model does not always lead to the maximization of interpretability in FRBS. A longer rule may be more meaningful than a shorter (more compact) rule without explicit semantics embedded in the fuzzy rules. However, in the literature of multi-objective FRBS design, interpretability maximization is mostly handled as complexity minimization using complexity measures (often termed as readability), such as the number of fuzzy rules, the total number of antecedents (i.e., total rule length), and number of inputs for each rule. Based on the existing studies, different approaches, namely, (i) rule generation, (ii) rule selection, and (iii) tuning of fuzzy sets from existing FRBS are some options used by MOEA to achieve the complexity-accuracy trade-off. In [25], a three-objective evolutionary algorithms (EAs) was proposed to generate Mamdani FRBS with the trade-off among accuracy, complexity of the rule base (i.e., the number of rules in the rule base and number of linguistic terms in antecedents of rules), and partition integrity (i.e., measures how much the partition is different from an initial interpretable partition). The rule base and membership function parameters were learned concurrently during the evolutionary process. Casillas et al. [26] proposed a genetic fuzzy system for regression problems. This method dealt with the issues, such as lack of completeness, inconsistency, redundancy, or over-generalization, which are common when the fuzzy rules come in the conjunctive normal form as antecedent for a compact knowledge representation capability. Another method of achieving this trade-off is to improve the existing FRBS focusing on rule selection. MOEAs with this idea are proposed in the literature [27] and [28]. Indeed, the initial model is usually made of a high number of rules; then, a subset of rules is selected by removing redundant and inconsistent rules to represent a more compact fuzzy model. Ishibuchi et al. [29] constructed a compact fuzzy classification system by formulating a rule selection problem with two objectives: (i) maximization of the number of correctly classified training patterns and (ii)

minimization of the number of selected rules. Later, Ishibuchi and Nojima [27] employed a hybrid version of the Michigan [30] and Pittsburgh [31] approaches to examine the complexity-accuracy trade-off in fuzzy rule-based classifiers. Unlike Ishibuchi and Nojima [27], the complexity was measured by the number of fuzzy rules and/or the total number of antecedents of fuzzy rules. These measures worked well as a safeguard against the overfitting of fuzzy rule-based classifiers in training patterns. Lahsasna and Seng [32] aimed at improving the model proposed by Ishibuchi and Nojima [27]. They proposed two variants, namely (i) employed an enhanced version of non-dominated sorting GA II (NSGA-II), so called Controlled Elitism NSGA-II for accuracy-complexity trade-off optimization, (ii) improved the selection of the antecedents of the rules generated in the initial population of GA algorithm using a feature-based selection of the most important features instead of random selection. Márquez et al. [33] utilized adaptive defuzzification methods in linguistic fuzzy rules; weight associated with each rule was introduced to prune FRBS. In that method, the aim of interpretability was reducing the number of rules. Tuning of fuzzy sets from the existing FRBS, assumes a predefined rule base and has an objective of finding a set of optimal parameters for the membership functions. MOEA-based rule selection and tuning were used in the literature [34] and [35]. Gacto et al. in [35] proposed an index, namely GM3M that preserved the semantic interpretability of linguistic fuzzy models. Additionally, rule selection mechanisms can be used to reduce the model complexity. To achieve this, a MOEA is used for accuracy, semantic interpretability maximization, and complexity minimization. Saad, et al. in [36] proposed a robust structure identification method (RSIM) based on incremental partitioning learning. RSIM proceeded with an open region (initial domain) that covered all input samples. The initial region started with one fuzzy rule without fuzzy terms and then evolved through incremental partitioning learning, which created many sub-regions for system error

minimization. It located sufficient splitting points provided through a robust partitioning technique, determined the optimum trade-off between accuracy and complexity through a partition-selection technique, minimized global error through global least square optimization. This method sought to produce a few rules with a low number of conditions to reduce system complexity. A review of the application of MOEA for fuzzy systems is available in the literature [37-38].

However, most of existing MOEA for the development of fuzzy models generate all the fuzzy rules in the same feature space, which is analogous to making decision based on the view of only one expert, yet admitting variability of input spaces in the antecedents of different rules as well as variability of orders in the local functions forming the consequences allows the fuzzy model possesses a more human-like inference mechanism; in other words, that is analogous to making a decision based on the views of different experts. In the model proposed in chapter, this variability is provided by an architecture composed of integration of PSO and LSE methods.

3.2. Problem formulation

In this dissertation, two structural arrangements for the rules with the intent of complexity management are used, namely, (i) structuralization of local functions forming consequences of the rules and (ii) structuralization of the input spaces of the antecedent parts.

Structuralization of local functions forming consequences of the rules: The order of each function ranges from 0 (a constant function) to 3. Thus, the cumulative order of c rules is pc, where p assumes values from 0 to 3. In other words, we have

$$order_1+order_2+\ldots+order_c= pc \tag{3.1}$$

where the $order_i$ is the order of i^{th} rule, c is the number of rules, and the parameter p is used to adjust the cumulative order of polynomials forming the consequence of the rules. If p =0, this implies that all consequences are just constant functions. If p =3, this means that all rules have consequences in the form of the polynomials of order 3. There are values of p located in-between when there are rules of consequences exhibiting orders between 0 and 3.

The objective of structuralizing local functions is forming a fuzzy rule-based model that involves the allocation of orders to the local polynomials of individual rules in such a way that the performance index of the model (e.g., the RMSE) becomes minimized, and the total order of polynomials across all the rules is constrained to pc. Assuming that $\hat{y}_k$ is the result produced by the fuzzy rule-based model for k^{th} input, the RMSE index Q is expressed as follows:

$$Q = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k-\hat{y}_k)^2} \tag{3.2}$$

Structuralization of the input spaces of the antecedent parts: The cumulative dimensionality of the input spaces of all rules (viz. the total number of input variables over all rules in the original format)) is $n+n+\ldots+n= n\times c$, where n is the original dimensionality of the input space. We admit variability of these spaces across different rules, ranging from n to $n\times c$. To model this relationship, we introduce reduction parameter β which satisfies the relationship

$$n_1+n_2+\ldots+n_c = \beta n \tag{3.3}$$

where n_1, n_2, …, and n_c are the numbers of variables in the corresponding rules. The values of β positioned in the range [1, c] help capture the range of admissible values of the dimensionalities of the input spaces. Structural arrangements of the input spaces in the antecedents are aimed at

selection of the most influential input variables of individual rules whose total dimensionality is constrained to βn.

3.3. Proposed architecture for optimizing the structural refinements of rules

We develop a hybrid approach based on PSO, information granulation (e.g., Fuzzy clustering), and the standard LSE to form a fuzzy rule-based model based on the resources already discussed (p, c, and β) in Section 3.2. The overall architecture of this methodology is shown in Figure 3.1. In this architecture, three main modules are responsible for forming the fuzzy model directly from input/target data. These components are as follows: (i) the FCM, which is responsible for the creation of the fuzzy sets in the antecedent parts of rules, (ii) the optimization engine, which is composed of PSO and standard LSE (The PSO is used to achieve the optimal arrangements of the input spaces in antecedent parts and the orders of polynomials in consequences, while the LSE is in charge of estimating the optimal coefficients of polynomials), and (iii) the evaluation module which is employed to calculate the cost function, viz, the RMSE of the fuzzy model over the evolution process.

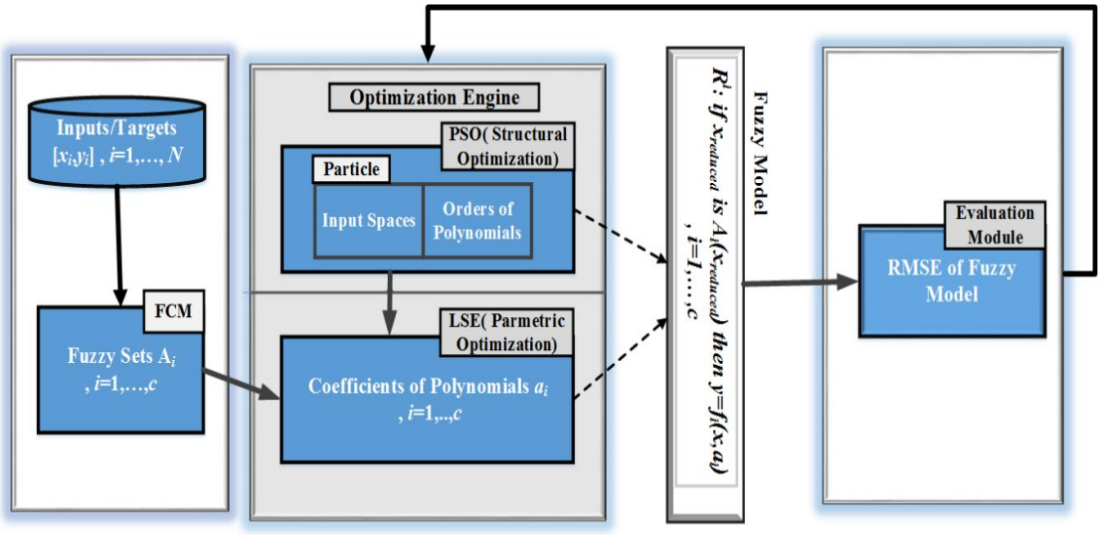


Figure 3. 1. The overall scheme of the proposed design methodology of FRBS

In the fuzzy model, suppose that the linear polynomials in the rules' consequences employ the original input space, while antecedents use subsets of input space (e.g., the reduced input space). Fuzzy rules are modified in the following form.

$$R^i: \text{ if } \boldsymbol{x}_{reduced} \text{ is } A_i(\boldsymbol{x}_{reduced}) \text{ then } y=f_i(\boldsymbol{x},\boldsymbol{a}_i) \quad , i=1, 2,\ldots, c \tag{3.4}$$

where $\boldsymbol{x}_{reduced}$ is a reduced subset of input space in the antecedent of i^{th} rule. However, to preserve accuracy, the input space in the consequent part retains the original $\boldsymbol{x}$.

The degree of activation of each rule is calculated based on the reduced input data, while the local function's value is computed from the original data. Thus, the activation level is computed as follows:

$$A_{ik}=\frac{1}{\sum_{j=1}^{c}\left(\frac{\| \boldsymbol{x}_{reduced_k}-\boldsymbol{v}_{reduced_i}\|}{\|\boldsymbol{x}_{reduced_k}-\boldsymbol{v}_{reduced_j}\|}\right)^{\frac{2}{m-1}}} \tag{3.5}$$

where $\boldsymbol{x}_{reduced_k}$ and $\boldsymbol{v}_{reduced_i}$ are respectively k^{th} data in the reduced space and the corresponding prototype of the i^{th} cluster (rule). While calculating optimal coefficients of local functions, the whole process is the same discussed in chapter 2 for the standard fuzzy rules. However, there are two changes regarding the notation: (i) symbol $\boldsymbol{x}$ is replaced with $\boldsymbol{x}_{reduced}$ only in the antecedents of rules, and (ii) the polynomials on the consequences of the rules can vary to constant, linear, quadratic, or cubic, despite the standard TSK model in which all the local functions are of the same type. This variable degree of polynomials in the new model resulted from the structural optimization process by the PSO.

The concept of a particle in the context of structuralization of local functions and input spaces to create fuzzy rules can now be discussed. Three different alternatives are proposed in this work to form optimized fuzzy rule-based models. The first model is built based on only optimizing the

arrangement of input variables in the antecedents of rules, while all the rules have the same orders of polynomials. The second model is created based on only optimizing the arrangement of orders of the polynomials forming consequences; however, the input spaces of antecedents retain the original input space. The last model is formed based on the optimization of both the antecedents and consequences of rules.

As the PSO is carried in the continuous search space, the contents of each particle are real-valued. The velocity and position values are supposed between zero and one in the design of our fuzzy model. The size of each particle (i.e., fuzzy model) in the third alternative is $c \times (n+1)$, where c is the number of rules and n is the original input space's dimension. However, the position size in the first and second alternatives are $c \times n$ and c, respectively.

We are focusing on the third model as it is a generalization of the first and second models. The scheme of coding candidate solution (i.e., particle position) is visualized in Figure 3.2.

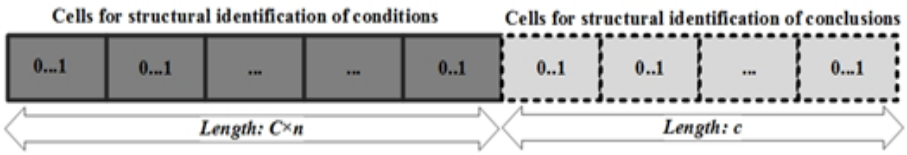


Figure 3. 2. Scheme of coding candidate solutions

The contents of cells are real-valued in the [0, 1] interval. Each solution is composed of two parts: the first part is used for structural identification of antecedents, and the second part is employed for structural identification of consequences. Thus, it is transformed over two steps into (i) reduced input spaces and (ii) the orders of corresponding polynomials.

In step one, the first $c \times n$ values of the candidate solution are transformed into subsets of input variables standing in the antecedents of rules. Algorithm 3.1 shows the pseudocode to form subsets of variables in each rule.

Algorithm 3.1. Decoding the antecedent of rules

Input: Candidate solution, number of rules (c), original dimension of input space (n), Reduction Parameter (β)

Output: A two-dimensional list S composed of the rule's number (#rule) and its corresponding variables' number (#variable)

1: S=[]
2: Sort the first $c \times n$ values of the candidate solution in ascending order
3: Retrieve the first $\beta \times n$ sorted indices of the candidate solution
4: **foreach** *index* in the retrieved indices
5: $\#rule = \left\lceil \frac{index}{n} \right\rceil$
6: **if** *index* mod $n == 0$
7: $\#variable = n$
8: **else**
9: $\#variable = index \bmod n$
10: **end if**
11: append *#variable* into S[*#rule*]
12: **end for**
13: **return** S

Take for example a rule-based model with c=3, n=4, $\beta=\frac{3}{2}$ and $p=\frac{5}{3}$. Suppose that the content of the candidate solution is as shown in Figure 3.3. Ranking the 12 elements of the solution leads to the order of indices [10, 2, 8, 1, 12, 4, 7, 3, 5, 11, 6, 9]. With $\beta=\frac{3}{2}=\frac{c}{2}$ (i.e., half of the original input

space), the selected indices are [10, 2, 8, 1, 12, 4]. As the size of the original input space is n=4, the reduced input spaces are as follows:

1th rule: x_1, x_2, x_4

2th rule: x_4

3th rule: x_2, x_4.

The last c values of the candidate solution are utilized in the second step to form the orders of polynomials. Given the total order of $p \times c$, we normalize the values so that their sum becomes $p \times c$. The normalized values may need to be rounded off to the nearest integer. For the particle's position mentioned in Figure 3.3, $p \times c$ =5 makes the polynomials in the first, second, and third rule quadratic, linear, and quadratic, respectively.

Figure 3. 3. A representation of particle's position in the structural optimization of FRBS

Note that the process of optimization for both first (only optimized antecedent) and second alternatives (only optimized antecedent) is the same as the third option explained above, whereas there is no structuralization of the orders in the first model and no structuralization of the input spaces in the second model.

Two commonly encountered requirements to retain the semantics of fuzzy sets concern (i) coverage of the input space and (ii) distinguishability of fuzzy sets. Regarding (i), this antecedent is met given the constraint imposed by the FCM algorithm, namely

$$\sum_{i=1}^{c} A_i(\boldsymbol{x})=1 \tag{3.6}$$

that is, for any $\boldsymbol{x}$ at least one rule (cluster) becomes active. In the case of (ii), the fuzzy sets' distinguishability is related to the prototypes' closeness (modal values of the fuzzy sets). Two fuzzy sets positioned too close to each other entail a significant degree of overlap, reducing the semantics of the rules (implies their redundancy). This effect can be controlled by limiting the number of rules, viz. the number of clusters, by imposing the following requirement:

$$\min_{i,j=1,2,\ldots,c} \|\boldsymbol{v_i}-\boldsymbol{v_j}\| \geq t \tag{3.7}$$

where $\boldsymbol{v_i}$ stands for the prototype of i^{th} cluster and t is a certain threshold. That is, the number of rules should not exceed the upper limit c_{max} beyond which the smallest distance between the prototypes is not lower than t.

3.4. Experimental studies

Numerical instances for the performance evaluation of the proposed approaches are provided here. Synthetic data, either one-dimensional or multi-dimensional, and some regression datasets from the UCI Machine Learning Repository, the Bilkent University Function Approximation Repository, and the KEEL Datasets are utilized in this chapter. In this section, we investigate the impact of structuralizing the antecedent and consequence of the rules on the model performance. There is a possible interest to determine whether (or not) the arrangement of antecedents is more efficient than the arrangement of consequences. Thus, four different alternatives in this work are compared to find a possible answer to the question raised here.

For the model formed based on one-dimensional data, only the consequent part is optimized, whereas, for multi-dimensional data, both the antecedent and consequences are optimized. In the

experiments, we use a 10-fold cross-validation method. The data set is divided into ten subsets of equal size. For each case, a subset is selected as the test dataset, and the remaining part of the data serves as the training dataset used to construct the model.

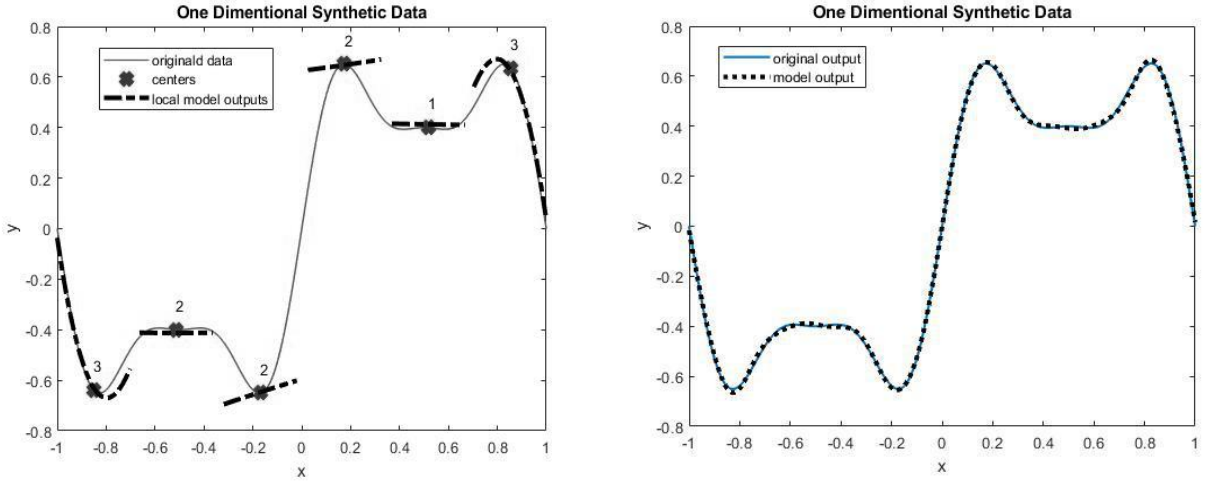
3.4.1. One-dimensional synthetic data

We consider a single-variable non-linear function of the following form $y=0.6 \sin(\pi x) + 0.3 \sin(3\pi x) + 0.1 \sin(5\pi x)$, where input x is defined in the [-1, 1] interval. 500 pairs of input-output data are generated by the given function y, where the inputs are evenly spaced points in the range [-1, 1]. Because there is only a single input variable, there is no need to reduce input space; thus, the PSO is used to distribute only polynomials' orders.

In this experiment, the modeling information is supposed to be as follows: $c=6$, $p=\frac{13}{6}$ (i.e., the total order of polynomials=13). For the PSO and FCM algorithms, the following parameters after a fine-tuning have been chosen: swarm size=50, the maximum number of iterations=50, inertia weight damping ratio=0.99, $w=1$, $c_1=2$, $c_2=2$, and the fuzzification coefficient as $m=1.9$.

The allocation of orders achieved by the proposed approach is displayed in Figure 3.4.a. This figure visualizes the prototypes made by the FCM algorithm and the outputs of each local model in a subspace. The values written on top of the prototypes indicate the orders which are allocated to the local functions. Local models can predict outputs relatively well in the given neighborhood; in other words, this leads to forming a locally well interpretable model. Indeed, local models express good interaction with the global model, and each polynomial depicts nicely the characteristics of the input-target relationship in the corresponding subset.

Figure 3.4.b illustrates the overall output of the proposed model on the given data. Based on the figure, the model output is usually similar to the original output. The overall performance index (RMSE) achieved by the model is 2.2925e-04.



(a) (b)

Figure 3. 4. Output of the model on one-dimensional data: (a) local models and their corresponding prototypes (b) overall output of the model

3.4.2. Multi-dimensional synthetic data

A 5-D random input data $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{500}$ is generated over [-10, 10] using a uniform distribution. Six prototypes in input space are formed, and then five linear polynomials and one constant are created as local functions of individual rules. The rules are formed as follows:

R^1: If $\boldsymbol{x}$ is A_1 with $\boldsymbol{v_1}$=[x_2=-0.59, x_3=0.55, x_5=0.52] then

f_1=-0.91-0.26x_1+0.02x_2-0.72x_3+0.27x_4-0.31x_5

R^2: If $\boldsymbol{x}$ is A_2 with $\boldsymbol{v_2}$=[x_2=0.08,x_3=-0.59, x_5=0.55] then

f_2=-0.26-0.72x_1+0.98x_2+0.03x_3-0.66x_4-0.91x_5

R^3: If $\boldsymbol{x}$ is A_2 with $\boldsymbol{v_3}$ =[x_2=-0.59, x_4=0.28] then f_3=0.31+0.83x_1+0.77x_2+0.94x_3+0.36x_4-0.08x_5

R^4: If $\boldsymbol{x}$ is A_4 with $\boldsymbol{v_4}$=[x_1=-0.09, x_2=-0.59] then f_4=-0.28

R^5: If $\boldsymbol{x}$ is A_5 with $\boldsymbol{v_5}$=[x_1=0.08, x_2=-0.59, x_4=0.27, x_5=0.52] then

f_5=0.26+0.87x_1-0.24x_2-0.69x_3-0.92x_4-0.18

R^6: If $\boldsymbol{x}$ is A_6 with $\boldsymbol{v_6}$=[x_3=0.55] then f_6=-0.74

Following that, the outputs of synthetic data are generated using the abovementioned rules and (2.9). The generated data, along with the system's resources (c=6, p=1, and β=3; i.e., cumulative dimensionality is 50% of the original input spaces) are then employed to develop four different fuzzy models.

In the first model, the original input space is preserved in antecedents of rules, and the polynomials in the consequences of all the rules are of the same order. However, for the rest of the three models, the PSO is used to optimize the antecedent or consequence of rules. That is, the second model is built based on optimizing the arrangement of orders in the polynomials, while the third model is created based on optimizing the arrangement of input variables in the antecedents. The last model is formed based on the optimization of both the antecedent and consequence. In this experiment, the parameters of PSO and FCM are chosen as: swarm size=50 (10n), maximum number of iterations=50 (10n), inertia weight damping ratio=0.99, w =1, c_1=2, c_2=2, and m=2. Performance comparison of the models is detailed in Table 3.1.

Table 3. 1. Performance comparison of four models on synthetic data (c=6, p=1, and β=3)

	No optimization	**Optimized consequence**	**Optimized antecedent**	**Both optimized antecedent and consequence**
Training data	0.0075±0.0017	0.0020±0.0001	0.0045±0.0032	**0.0018±0.0021**
Testing data	0.0018±0.0221	0.0023±0.0011	0.0031±0.0025	**0.0012±0.0014**

Among the models illustrated in Table 3.1, the model with both the optimized antecedent and consequence performs better than others regarding the accuracy criterion. The model with the only optimized consequence has a relatively similar performance; that is, it is seen that the values of the performance indices achieved by the only optimized consequence, particularly for the training data, are very close to the performance of the model with both the optimized antecedent and consequence. The complexity of rules in the model with the antecedent and consequence both optimized are visualized in Figure 3.5. The values of the pairs ($order_i$, n_i) in this figure demonstrate the complexity of the i^{th} rule, wherein $order_i$ and n_i are the order of polynomial and the dimensionality of input space in the i^{th} rule, respectively.

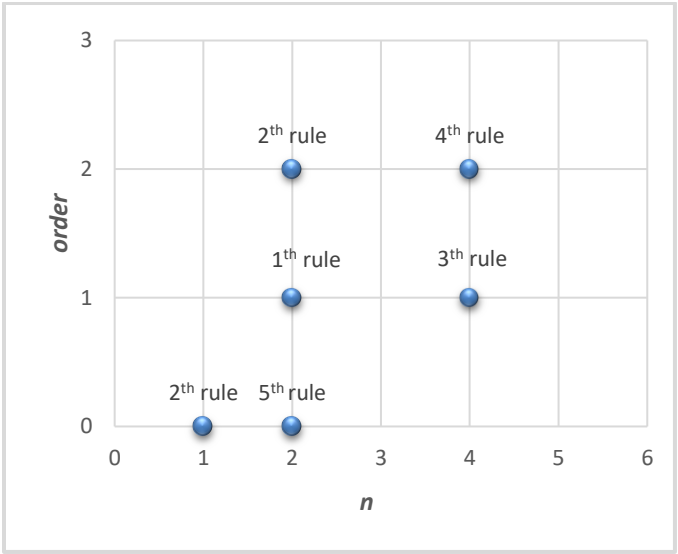


Figure 3. 5. The complexity of rules created based on synthetic data (with β=3)

In the next experiment, keeping all the parameters fixed, the value of β changes to 4.8 (i.e., 80% of the original input space); the performance comparison results are illustrated in Table 3.2. This table indicates a quite similar tendency to Table 3.1; here, optimization in both the antecedent and consequence also leads to superior performance on the training data. Optimization on only the

consequence also achieves better results, particularly on the testing data, than optimization on only the antecedent or no optimization. Thus, the models formed by these two optimization processes (both optimized antecedent and consequence, and only optimized consequence) not only indicate the improvement of complexity of the models (due to input space reduction and allocation of the low orders to the polynomials), but also help in retention of approximation abilities. The complexity of the fourth model is displayed in Figure 3.6.

Table 3. 2. Performance comparison of four models on synthetic data (c=6, p=1, and β =4.8)

	No optimization	**Optimized consequence**	**Optimized antecedent**	**Both optimized antecedent and consequence**
Training data	0.0079±0.0018	0.0018±0.0005	0.0040±0.0004	**0.0005±0.0001**
Testing data	0.0270±0.0015	**0.0019±0.0001**	0.096±0.0021	0.0030±0.0013

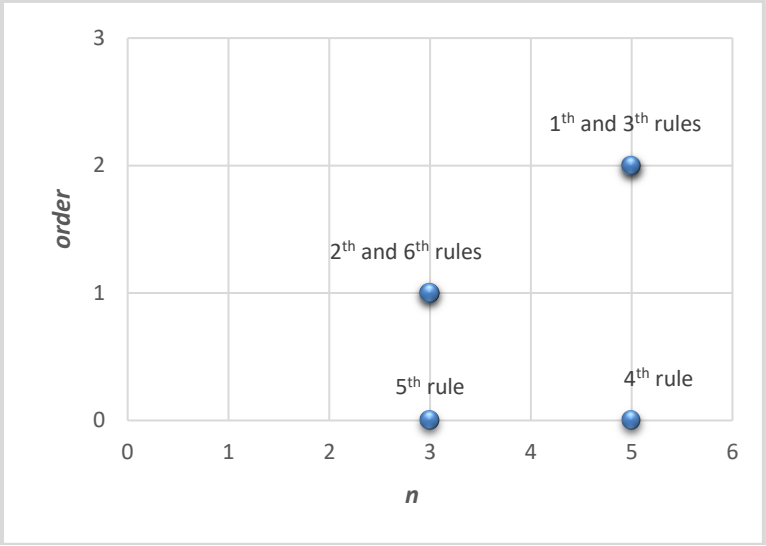


Figure 3. 6. The complexity of rules created based on synthetic data (with β=4.8)

The comparison of the results included in Table 3.1 and Table 3.2 reveals an interesting relationship whereby the increase of β for the generated synthetic data leads to the outputs with a performance index that is quite less. This improvement of performance is intuitively appealing;

one possible reason is that presence of more subset of inputs increases the accuracy of model. This higher β value becomes more vital, especially when the original input space is of the low dimensionality.

3.4.3. Publicly available datasets

Several datasets, with different sizes and dimensions, from the UCI Machine Learning Repository, the Bilkent University Function Approximation Repository, and the KEEL Datasets are employed to evaluate the performance of the models. The first dataset, the Yacht Hydrodynamics, is used from the UCI. This dataset comprises 308 instances with six input variables and a single continuous output. The residuary resistance of sailing yachts per unit weight of displacement is the output of the model. The experiments are performed with different values of β, *e.g.,* β=1.8, 3, and 4.8. The number of rules is set to 6, and the total order to 6 (e.g., p=1). The weighted Euclidean distance is used as the distance function in our experiment. Four different fuzzy models mentioned in Section 3.4.1 are performed on the Yacht Hydrodynamic dataset, and the results are recorded in Table 3.3. Parameters used for PSO and the fuzzification coefficient are like the ones employed for the multi-dimensional synthetic data. It is expressed in Table 3.3 that for the Yacht Hydrodynamics dataset, optimizing the arrangement of orders standing in the consequences leads to a more accurate model than optimizing the input spaces in the antecedents. The possession of the whole input variables and the proper allocation of orders achieved through the optimization process, instead of the uniform allocation of orders, gives the possibility of increasing the accuracy of the model compared with possession of the reduced (optimized) input spaces. However, optimization on both the antecedent and the consequence is still a promising alternative in the given dataset. This optimization creates a model which is accurate enough yet more compact, in particular when β=1.8.

Table 3. 3. Performance comparison of four models on the Yacht Hydrodynamics dataset (c=6, and p=1)

Dataset (Size)	β		No optimization	Optimized consequence	Optimized antecedent	Both optimized antecedent and consequence
Yacht Hydrodynamics (308*7)	4.8	**Training data**	2.23±0.17	**0.55±0.11**	1.10±0.21	0.61±0.12
		Testing data	2.78±0.47	**0.94±0.12**	1.41±0.24	1.11±0.56
	3	**Training data**	2.24±0.23	**0.53±0.15**	1.30±0.14	0.65±0.08
		Testing data	2.85±0.32	1.16±0.21	1.43±0.11	**1.12±0.09**
	1.8	**Training data**	2.23±0.12	**0.56±0.13**	1.38±0.09	0.73±0.15
		Testing data	2.76±0.16	**1.11±0.12**	1.25±0.16	1.16±0.14

The second dataset from the UCI is Boston Housing, which concerns housing values in Boston's suburbs. This dataset includes 506 instances where the input space is 13, and the output is a continuous singleton. The performance indices of the four different models are shown in Table 4. The model that uses the whole original space and optimizes order allocation obtains superior results compared to the other models. Table 3.4 illustrates that optimizing order allocation for this dataset performs better than optimizing input spaces.

Table 3. 4. Performance comparison of four models on the Boston Housing dataset (c=6, p=1)

Dataset (Size)	β		No optimization	Optimized consequence	Optimized antecedent	Both optimized antecedent and consequence
Boston Housing (506*14)	4.8	**Training data**	2.82±0.23	**1.04±0.08**	2.52±0.16	1.20±0.08
		Testing data	2.70±0.10	**1.32±0.21**	2.23±0.18	1.63±0.17
	3	**Training data**	2.87±0.12	**1.07±0.20**	2.60±0.23	1.35±0.14
		Testing data	2.94±0.25	**1.36±0.14**	2.53±0.13	1.81±0.07
	1.8	**Training Data**	2.84±0.18	**1.06±0.12**	2.68±0.13	1.55±0.17
		Testing data	2.80±0.33	**1.66±0.13**	2.88±0.32	1.76±0.24

The concrete compressive strength is another dataset from the UCI that is employed in our experiments. This dataset is a highly nonlinear function of age and ingredients and comprises 1030

instances with nine input variables and one real output. Like previous datasets, the models' performance indices are measured for the concrete compressive strength dataset and displayed in Table 3.5. This table still indicates the dominance of the model with optimized consequences. Nevertheless, simultaneous optimization of the antecedents and consequences is also an encouraging method to create an interpretable yet accurate fuzzy model.

Table 3. 5. Performance comparison of four models on the Concrete Compressive Strength dataset (c=6, p=1)

Dataset (Size)	B		No optimization	Optimized consequence	Optimized antecedent	Both optimized antecedent and consequence
Concrete Compressive Strength (1030*9)	4.8	**Training data**	6.21±0.17	**4.84±0.13**	5.73±0.25	4.93±0.31
		Testing data	6.34±0.18	6.05±0.21	5.90±.14	**5.24±0.31**
	3	**Training data**	6.19±0.24	**4.86±0.21**	5.96±0.19	5.33±0.14
		Testing data	6.10±0.17	5.89±0.15	5.95±0.24	**5.55±0.12**
	1.8	**Training Data**	6.23±0.14	**4.80±0.09**	6.14±042	5.41±0.24
		Testing data	6.27±0.20	6.16±0.31	5.92±0.12	**5.67±0.31**

Further datasets from the UCI, Bilkent, and KEEL repositories are utilized to compare the effectiveness of the arrangement of input spaces in antecedents and polynomials' orders in consequences. Experiments are performed with β=4.8, c=6, and p=1. The ratio of $\frac{Q'}{Q}$ in Table 3.6 is recorded for the optimized models in testing and training data. Q' and Q are, respectively, the performance indices achieved by the optimized model and the model in which orders of all the polynomials are of the same type. For the training data, in 53% of the cases, optimization in consequences contributes to better results, while this value decreases to 10% when the optimization is only on the antecedents of rules. This information stresses that the design of fuzzy rules based on allocating orders of polynomials performs mostly better than the reduction of input space with respect to the accuracy.

Table 3. 6. Performance Comparison of the optimized models

Dataset	Size	$\frac{Q}{Q'}$		
		Optimized consequence	**Optimized antecedent**	**Both optimized antecedent and consequence**
Forest Fires	517*13	0.72±0.11	0.83±0.13	**0.52±0.08**
		0.93±0.14	1.18±0.08	1.06±0.13
Combined Cycle Power planet	9568*4	**0.93±0.12**	1.35±0.10	1.12±0.16
		1.00±0.06	1.80±0.12	1.04±0.13
Auto MPG	398*8	0.94+0.07	0.98±0.14	**0.85±0.07**
		0.96±0.13	1.02±0.13	0.97+0.10
Airfoil Self-Noise	1503*6	1.00±0.05	0.97±0.17	**0.95±0.08**
		1.00+0.11	0.98±0.15	1.00±0.11
Mortgage	1049*16	**0.90±0.07**	0.98±0.17	0.97±0.13
		1.00±0.06	0.97±0.13	0.97±0.09
Treasury	1049*16	0.98±0.14	0.98±0.17	**0.97±0.09**
		1.00+0.09	1.11±0.14	1.09±0.13
Weather Izmir	1461*10	**0.91±0.04**	1.15±0.10	0.97±0.18
		0.95±0.10	1.18±0.16	1.08±0.14
Weather Ankara	1609*10	**0.91±0.07**	1.03±0.12	0.98±0.10
		1.26±0.10	1.13±0.11	1.10+0.13
Villages	766*32	**0.85±0.08**	0.95±0.13	0.90±0.08
		0.15±0.17	0.42+0.10	0.33+0.10
Stock Prices	950*9	**0.72**±0.05	1.12±0.18	0.75±0.13
		1.03±0.11	1.09±0.12	2.33±0.18
Sleep	57*7	**0.01±0.00**	0.12±0.03	0.03±0.00
		0.40±0.04	0.58±0.09	0.25±0.06
PW Linear	200*10	0.02±0.00	0.62±0.06	**0.00±0.00**
		1.00±0.12	1.45±0.12	1.00±0.10
Pollution	60*15	0.90±0.15	**0.17±0.02**	0.20±0.03
		1.05±0.13	0.95±0.13	0.41±0.11
Northridge Earthquake	2929*10	0.95±0.14	0.93±0.10	**0.85±0.07**
		1.10±0.12	0.94±0.14	0.94±0.15
Elevators	16599*18	1.00±0.06	**0.97±0.12**	**0.99±0.09**
		1.00±0.13	0.99±0.10	0.99±0.13
Pole Telecomm	9065*48	**0.90±0.17**	0.97±0.16	0.95±0.09
		0.93±0.21	0.98±0.18	0.98±0.13
2Dplanes	40768*10	**0.93±0.03**	1.05±0.15	0.91±0.08
		0.95±0.10	1.13±0.17	0.98±0.14
Ailerons	13750*40	**0.90±0.10**	0.95±0.04	0.92±0.08
		0.97±0.05	0.98±0.08	0.97±0.14
Pumadyn	8192*32	0.91±0.10	0.91±0.09	**0. 87±0.03**
		1.00±0.12	1.21±0.11	1.01±0.13

Note: the entities in boldface represent the best performance obtained for two methods

Different clustering schemes in the literature can be adopted in the TSK model design, e.g., subtractive clustering [62], Gustafson Kessel clustering [63]. As subtractive clustering is primarily dependent on the clusters' radius, not the number of clusters, it is difficult to be employed for the TSK model's structural arrangement based on the modeling resources (e.g., the total number of orders across all the rules; i.e., $p \times c$). Gustafson-Kessel algorithm is used as an alternative to form the antecedents of fuzzy rules. This algorithm uses the correlation of the variables and, consequently, the Mahalanobis distance instead of Euclidean distance, so it needs to calculate the inverse of the Covariance matrix, leading to an increase in the algorithm's cost. Each iteration of Gustafson-Kessel algorithm is $O(cNm^2)$ compared to the $O(cNm)$ for FCM, where c, N, and m are the number of clusters, the number of data items, the dimensionality of the dataset, respectively. We are forming TSK models in which both antecedents and consequences are optimized.

Table 3.7 illustrates the performance index (PI) and running time (in seconds) of TSK models (both optimized antecedent and consequence models), whose antecedents are built upon the FCM or Gustafson Kessel schemes. Table 3.7 clearly shows that the PIs of both models are very close to each other for different datasets, but the running time for the TSK design by FCM is lower than the one for TSK by Gustafson Kessel scheme. Thus, the low time complexity is the reason why we have chosen FCM in our experiments.

Table 3. 7. FCM vs Gustafson Kessel clustering

Dataset		**FCM**		**Gustafson Kessel**	
		PI	**Time(s)**	**PI**	**Time(s)**
Boston Housing	**Training data**	1.20±0.08	92.73±2.15	1.22±0.15	100.63±3.10
	Testing data	1.63±0.17		1.60±0.14	
Yacht Hydrodynamics	**Training data**	0.61±0.12	69.13±3.76	0.63±0.14	82.24±4.34
	Testing data	1.11±0.16		1.14±0.09	
concrete compressive strength	**Training data**	4.93±0.31	165.32±4.78	4.90±0.24	187.18±5.32
	Testing data	5.24±0.31		5.20±0.20	

3.4.4. Comparative analysis

Finally, Table 3.8 provides the comparative analysis of the proposed model with those existing in the literature. The experiments are conducted on a 1.30 GHz core i7 PC with 16 GB of RAM under the Matlab environment. PSO with only optimized consequence is used as the representative of the proposed models in this study.

Along with the use of PSO as the optimization vehicle, we experiment with GA that adopts the same scheme of coding candidate solutions that PSO in Section 3.3 uses. The proposed models are compared with two different GAs from the literature, the GA with binary encoding [15] and the GA with real-encoding [19]. To arrive at a sound comparative framework, the numbers of generations and the population's size are kept the same as those being used in the case of PSO. We intend to compare the quality of results produced by the different methods and look at the computational effectiveness of the methods themselves, so we record minimum PI and its running time (in seconds) for each model in Table 3.8.

Table 3. 8. Results of comparative analysis

Dataset		Binary GA [15]		Real-encoded GA [19]		proposed GA		proposed PSO	
		PI	**Time(s)**	**PI**	**Time(s)**	**PI**	**Time(s)**	**PI**	**Time(s)**
Boston Housing	**Training data**	2.46±0.23	120.78±4.5	3.13±0.31	100.63±3.33	1.06±0.10	98.13±2.10	**1.04±0.08**	80.13±3.22
	Testing data	2.60±0.18		3.17±0.34		1.27±0.17		1.32±0.21	
Yacht Hydrodynamics	**Training data**	2.22±0.13	100.12±3.3	1.10±0.13	83.36±2.94	0.57±0.08	78.10±2.02	**0.55±0.11**	60.45±2.67
	Testing data	2.79±0.18		1.96±0.22		0.91±0.10		0.94±0.12	
concrete compressive strength	**Training Data**	6.36±0.20	220.64±5.1	**4.32±0.12**	190.14±5.22	4.88±0.10	182.37±3.29	4.84±0.13	150.36±4.16
	Testing data	6.94±0.18		5.94±0.14		6.22±0.14		6.05±0.21	

Note: the entities in boldface represent the best performance obtained for two methods

We note that the results produced by the proposed PSO outperform both the GAs in the literature in terms of approximation capabilities and computational time (primarily due to the intrinsic simplicity of PSO) for the Boston Housing and Yacht Hydrodynamics datasets. In the concrete comprehensive strength dataset, real coding GA archives better performance than the proposed models, but its running time is more than the proposed models in this study. Furthermore, the proposed GA results in this study are not substantially different from the results of PSO, but from the computational point of view, PSO is still more efficient than GA. Thus, it is concluded that the proposed model based on PSO achieves a balanced performance index and running time.

3.5. Conclusions

In this chapter, an identification framework for fuzzy rule-based models has been developed. This framework stresses the need for and benefits of structural refinement of fuzzy rules to make the model less complex while still retaining its accuracy. Two different ways of structuralizing the antecedents and consequences of the rules, based on the newly introduced modeling resources, are proposed: (i) the arrangement of input spaces in the antecedents of the rules and (ii) the arrangement of the orders of polynomials in the consequences of the rules. The modeling resources are concerned with (i) the total order of polynomials encountered across all the rules and (ii) the fraction of an overall number of input variables of the original space. A hybrid methodology is proposed in which the PSO guided by the RMSE accuracy criterion is employed to find the efficient arrangements of input spaces and polynomials' orders. In this method, fuzzy sets standing in the antecedents of rules are created by the FCM, while the polynomials' coefficients are estimated by the standard LSE method.

Different TSK models formed in this study are based on the optimization of antecedents or consequences of fuzzy rules. The experimental studies, involving synthetic datasets and some well-known datasets from the UCI, Bilkent, and KEEL repositories, demonstrate the dominance of the model with optimal allocation of orders of polynomials in consequences over the reduction of input space in antecedents of fuzzy rules in most of the datasets. Based on the experiments, simultaneous optimization on both antecedents and consequences of fuzzy rules is also a promising avenue of creating fuzzy models for both complexity and accuracy criteria. The model's performance is also influenced by some crucial parameters of the model, such as the fraction of original input space employed in the model.

Chapter 4 Genetic-Programming Based Architecture of Fuzzy Modeling: Towards Coping with High-Dimensional Data

This chapter is concerned with developing fuzzy models realized with the aid of genetic programming (GP). The proposed architecture employs GP to form fuzzy logic expressions involving logic operators and information granules (fuzzy sets) located in the input space, used to predict information granules located in the output space. We propose an architecture realizing logic processing, with the structural optimization of the model accomplished by multi-tree genetic programming and the parametric optimization completed by gradient-based learning. The granulation of information used in this architecture is developed using the FCM clustering algorithm. The novelty of this study is two-fold: (i) it comes with the flexibility of the logic-oriented structure of fuzzy models, and (ii) our architecture is designed to handle high-dimensional data by alleviating the detrimental effect of distance concentration hampering the effectiveness of standard Takagi–Sugeno–Kang (TSK) fuzzy rule-based models. The study is illustrated through some experiments that provide a detailed insight into the fuzzy models' performance. A comprehensive comparative analysis is also covered.

4.1. High-dimensional data in the design of TSK models

One of the concerns related to high-dimensional computing is the phenomenon of distance concentration [7-8], where distances between the data items become very similar or identical. Thus, any machine learning algorithm which uses a distance function can be strongly affected by this issue; e.g., in FCM clustering, membership degrees of observations are proportionally

distributed across all the clusters. As the development of TSK models is fundamentally based on FCM, this phenomenon negatively impacts the model's performance. Motivated by this idea, this chapter's objective is to form more distinct and meaningful clusters in the design of the fuzzy models to cope with the issue of distance concentration. This is achieved by utilizing FCM independently to each variable rather than to all variables at once. We then form the fuzzy model employing GP as the logic processing core to form the model's structure, followed by the gradient descent to optimize parameters.

The research in evolutionary fuzzy systems (EFSs) has been experiencing rapid growth since its inception in the 90's. Generally speaking, an EFS [2] involves two aims: (i) parametric optimization and (ii) structural optimization.

After the pioneering work of Koza [12], GP gradually started to be used by the EFS's society due to its readable representation of fuzzy rules. Based on the literature concerning GP-based EFSs, the existing approaches can be categorized into two major groups: (i) grammar-based and (ii) symbolic-centered [5]. A context-free grammar is used in a grammar-based GP that allows forming fuzzy rules following a specific pattern (grammar). In general, the fuzzy rules developed by this approach are more transparent and can be incorporated into a fuzzy system as a slight adjustment is made in its reasoning mechanism. This group's representative model is the so-called GP-COACH [39], a compact classification model for high-dimensional problems. This EFS follows the genetic cooperative-competitive learning (GCCL) approach [5] that encodes a single rule per individual. However, the symbolic-centered methods employ the Koza-style GP, which uses a set of terminals and functions instead of a context-free grammar. The terminals are a set of features that have been already converted into fuzzy sets, and the functions are constrained to certain mathematical operators and serve as *t*-norms, *t*-conorms, and linguistic modifiers, such as

product, minimum, and square-root. A Koza-style GP was used in [41] to find a suitable structure of a fuzzy model for regression problems. In this fuzzy modeling method, the model's structure is determined by fuzzy neural networks, which are architectures governed by fuzzy logic. However, this model's logic-oriented structure is limited to basic fuzzy operators, specifically, the *t*-norms and *t*-conorms.

As mentioned earlier, high-dimensional data continues to pose challenges to the design of FRBS. Therefore, several approaches based on dimensionality reduction have been proposed in the design of FRBS to prevent the degeneration of such models' interpretability and accuracy. Feature extraction techniques, such as Principal Component Analysis (PCA), have been used to capture the critical components from the training data to enhance the interpretability of TSKs with concise fuzzy rules. For example, a research based on PCA was conducted in [42] to form a Type-2 hierarchical fuzzy system for handling high-dimensional data. However, the interpretability may be reduced since the physical meaning of the original features is lost. Feature selection is mainly used in the high-dimensional data-driven development of fuzzy models. In [43], GA and integer programming are integrated into the classification of high-dimensional data, in which the GA allows the absence of some input features in each rule. In this method, many rules are generated heuristically for each class, which are then selected to form a pool of rules using integer programming. However, the selection of features for the subsets is conducted quite randomly, which reduces the proposed model's effectiveness.

Another issue with most existing data-driven FRBS concerning high-dimensional data is that such models have been developed to generate all the fuzzy rules in the same feature space [44-46], which is not always compatible with human reasoning mechanisms. Using an elastic feature space is more appropriate for characterizing different views to better simulate the human reasoning

mechanisms. To achieve this aim, an elastic TSK fuzzy logic system (ETSK-FLS) [47] was proposed for the construction of TSK models based on high-dimensional data. In the ETSK-FLS, the soft subspace clustering (SSC) technique [48], [53] is first employed to determine the optimal partition of the input space and selection of the essential feature subsets for the antecedents of different rules. Then, the L2-norm penalty and structural risk minimization-based techniques are used to optimize the parameters of consequences. Indeed, the ETSK-FLS is primarily concerned with forming concise rules and creating a human-like reasoning mechanism for modeling systems based on high-dimensional datasets. However, the limitation of this approach is that the removal of redundant rules is not addressed. To deal with the issue mentioned for ETSK-FLS, a concise TSK construction method, called ESSC-SL-CTSK-FS [51], was proposed. In this method, the enhanced soft subspace clustering (ESSC) [49] is employed to create different sparse subspaces for the antecedents of fuzzy rules, while the sparse learning (SL) technique [50] was used to optimize the parameters of consequences of rules. Using the SL helps the construction method to reduce the number of fuzzy rules effectively in the model. Unlike most SSC algorithms that focus only on the within-cluster compactness, the ESSC presents an advantage as it considers not only the within-cluster compactness but also the between-cluster separation, which leads to creating rules which are more distinctive from each other.

To the best of our knowledge, the issue of distance concentration in high-dimensional data has not been discussed in any of the approaches mentioned above. Although those methods have been adopted for the high-dimensional data, the data dimensionality has often been limited to about 100 and fewer variables; that is, the trained fuzzy models' performance may deteriorate for real high-dimensional applications.

4.2. The architecture of the proposed fuzzy model and its underlying concepts

We develop a hybrid approach based on information granulation (fuzzy clustering), GP, and gradient descent algorithms. This architecture's motivation is the use of information granules (fuzzy sets) located in the input space and forming fuzzy logic expressions that predict the information granules located in the output space. In the sequel, the output information granules are used to generate numeric output through the process of degranulation (decoding, defuzzification).

In the proposed architecture, three primary components are recognized in the fuzzy model. These modules adhere to the standard fuzzy modeling architecture composed of: (i) input interface, (ii) logic processing core, and (iii) output interface. The overall architecture of the model is displayed in Figure 4.1.

Input interface (fuzzification) transforms the input data from a numerical level into the internal level of information granules understood by the logic-processing core. In this architecture, FCM, which is a representative clustering algorithm, forms the fuzzy sets. FCM is applied independently to each variable, rather than to all variables at once, to cope with the issue of distances concentration in high-dimensional data. This is used to split (partition) input and output data respectively into nc_1 and c_2 fuzzy sets, where n is the dimensionality of input space, c_1 and c_2 denote the number of clusters defined for a single input and the output space, respectively. The logic processing core (i.e., GP), the most crucial module of the fuzzy model, benefits from a multi-tree GP [40] to form an optimal fuzzy logic expression as the structure of the fuzzy model and details of system behavior. This module is looking for the optimal structure among the space of all possible model structures. The structure of the model is represented in the form of tree-like expressions composed of fuzzy operators and input fuzzy sets. Finally, the output interface (defuzzification) communicates the results of the processing module in a form acceptable by the

external world, viz., numeric values. The output interface is also equipped with gradient descent which is applied to add parametric flexibility to the model.

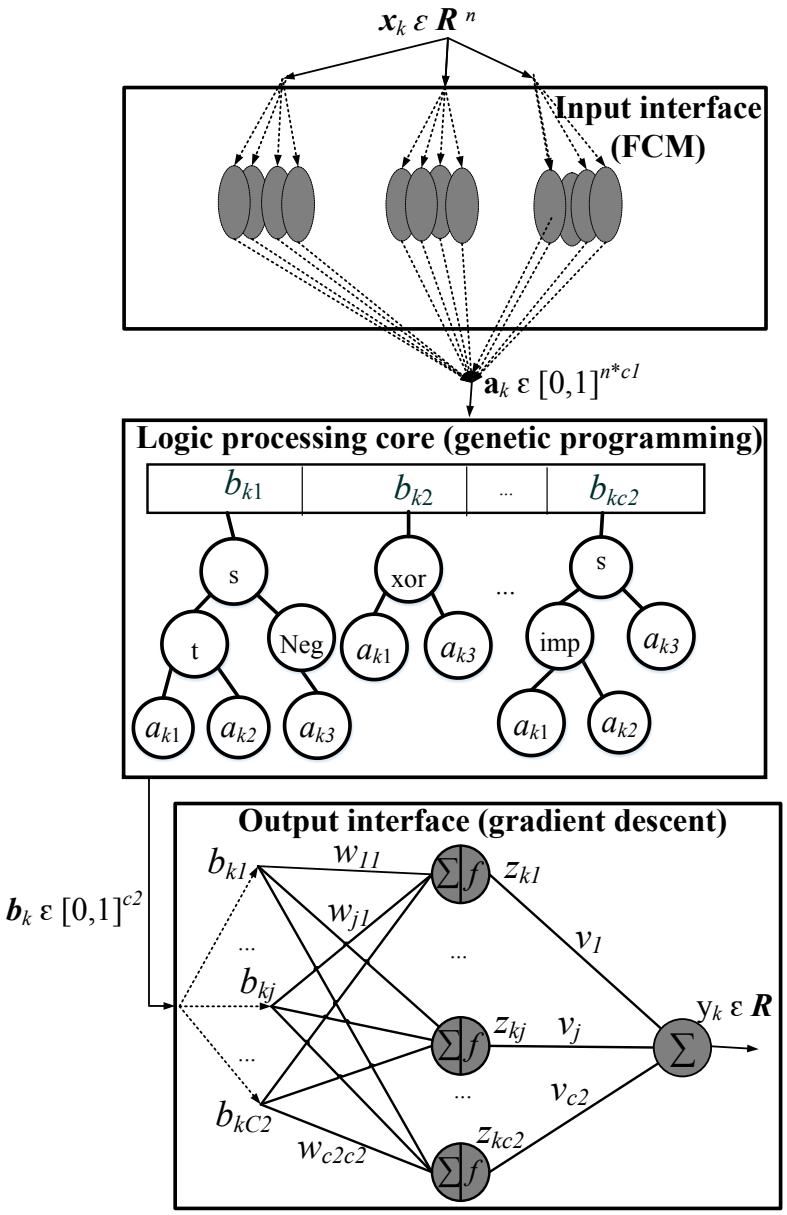


Figure 4. 1. The overall architecture of the proposed design methodology of the fuzzy model, with $c_1 = 4$, and $c_2 = 3$. Notation detailed in the text.

In detail, consider the data in the form of $(\boldsymbol{x_k}, target_k)$, k=1, 2, …, N, where $\boldsymbol{x_k}$=(x_{1k}, x_{2k},…, x_{nk})

and y_k is a numeric output. We require that y_k, the output of the fuzzy model for the input $\boldsymbol{x_k}$, be as close as possible to $target_k$; viz., $y_k \approx target_k$. In the first step of the development process, the training data, $(\boldsymbol{x_k}, target_k)$, $k=1,\ldots, N$, are converted through the input interface (FCM) into fuzzy sets, that is $(\boldsymbol{a_k}, \boldsymbol{t_k})$, where $\boldsymbol{a_k}$ denotes a vector composed of nc_1 input fuzzy sets formed from $\boldsymbol{x_k}$, and $\boldsymbol{t_k}$ denotes a vector of c_2 output fuzzy sets generated from $target_k$. In the next step, the GP is employed to map an input fuzzy set $\boldsymbol{a_k}$ to $\boldsymbol{b_k}$, where $\boldsymbol{b_k}$ is the output fuzzy sets generated by the GP. The GP module, as the processing core of the model, is trained to meet the requirement $\boldsymbol{b_k} \approx \boldsymbol{t_k}$. As Figure 4.1 shows, the structure of the model is represented as a collection of trees composed of input fuzzy sets and fuzzy operators. Fuzzy operators in these trees belong to a set of basic operators so-called *t*-norms (simply noted *t*), *t*-conorms (denoted as *s*), negations (Neg), and augmented operators such as Xor, Nor, Nand, Implication (Imp), and Equality (Eq). Finally, the decoding module, which is equipped with gradient descent optimization, is employed to represent the numeric output y_k of the fuzzy model. This module takes the output granules formed by the GP, say $\boldsymbol{b_k}$, then optimizes the weight connections (W) using the gradient descent in order to calculate the numeric output of the model, y_k. The design details of the proposed fuzzy model based on GP and gradient descent are given in Section 4.3.

4.3. Design/Optimization of fuzzy model

The identification of a fuzzy model can be formulated as a search problem in multidimensional space, where each point represents a potential fuzzy model with a given structure and related parameters. This section covers the optimization details of the proposed model.

4.3.1. Structural optimization of the fuzzy model with multi-tree GP

The processing core receives information granules formed from the numeric inputs by the input interface. The primary role of processing is to optimize a logic approximation of the experimental data. This is accomplished by building a model with the use of basic fuzzy logic operators (namely *t*-norm, *t*-conorm, and negation operations) and expanding them to the other operators, namely Xor, Nor, Nand, Implication, and Equivalence, which are tailored for fuzzy computing. We focus on the use of GP in this study as GP seems to have a straightforward and readable genotype-phenotype mapping for the structural optimization done here. In the following section, we will illustrate briefly the notion of GP and its application in structural optimization of the fuzzy model.

4.3.2. Genetic representation of the fuzzy model

The initial population is generated by the grow method [52] in which nodes can be randomly taken from both functions and terminals sets until the pre-defined depth limit is reached. Beyond that depth, only terminals can be chosen. Each individual in a population, viz. a single fuzzy model, is represented in a multi-tree form [40] composed of a set of tree structures; that is, each individual consists of c_2 tree structures that map $\boldsymbol{a} \in [0,1]^{nc_1}$ to $\boldsymbol{b} \in [0,1]^{c_2}$. The terminal set is composed of membership values in input spaces, whereas the function set consists of basic fuzzy operators, so-called *t*-norm (*t*), *t*-conorm (*s*) and negation (Neg). Combining the basic operators adds further diversity to the function set by forming additional operators in the function set. These additional operators are as follows; negation is expressed by the bar symbol (overbar).

x Xor $y = (x\ t\ \bar{y})\ s\ (\bar{x}\ t\ y)$

x Nand $y = (\overline{x\ t\ y})$

x Nor y= ($\overline{x\ s\ y}$)

x Imp y= ($\bar{x}\ s\ y$)

x Eq y=($\bar{x}\ s\ y$) t ($x\ s\ \bar{y}$)

Figure 4.2 illustrates an example of the representation of a fuzzy model with three outputs in the form of a multi-tree individual composed of the following expressions:

b_1= ($a_1\ t\ a_2$) s ($\bar{a}_3$); b_2= (a_1 Xor a_3); b_3= (a_1 Imp a_2) s (a_3), where a_1 to a_3, and b_1 to b_3 are input and output fuzzy sets, respectively.

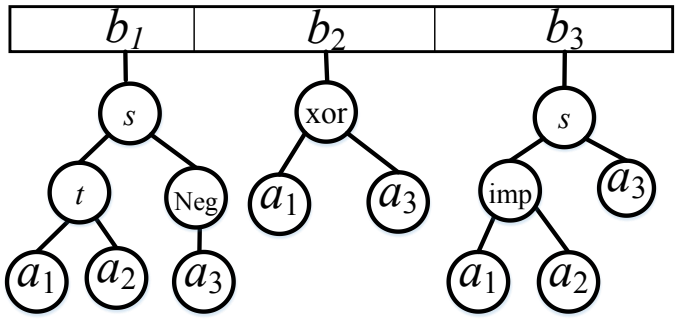


Figure 4. 2. Representation of the fuzzy model by a multi-tree individual.

4.3.3. Fitness function

A fitness Function determines how fit a solution is with respect to the problem in consideration. In our case, it assesses how well the information granules formed by the GP match the target information granules. This simply means that the optimization of the processing core (structural optimization) is guided by the performance available at the fuzzy model's internal level (level of the membership values). The internal level of accuracy quantification is schematically visualized in Figure 4.3. The fitness function, f, is determined in the following form,

$$f = -\sum_{k=1}^{N} (\boldsymbol{b}_k - \boldsymbol{t}_k)^{\mathrm{T}} (\boldsymbol{b}_k - \boldsymbol{t}_k) \tag{4.1}$$

where N is the number of training data, $\boldsymbol{b}_k$ is the output fuzzy set produced by the GP for the given data $\boldsymbol{x}_k$, and $\boldsymbol{t}_k$ is the target fuzzy set of $\boldsymbol{x}_k$.

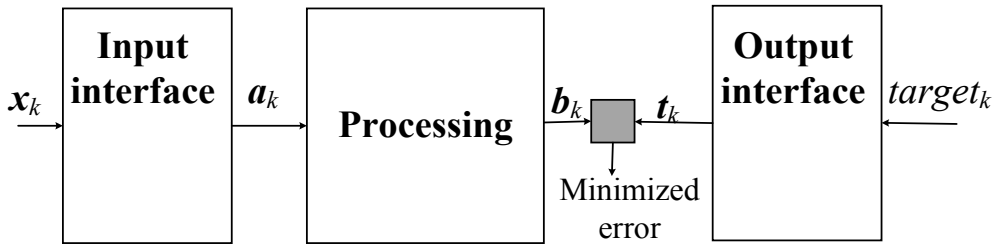


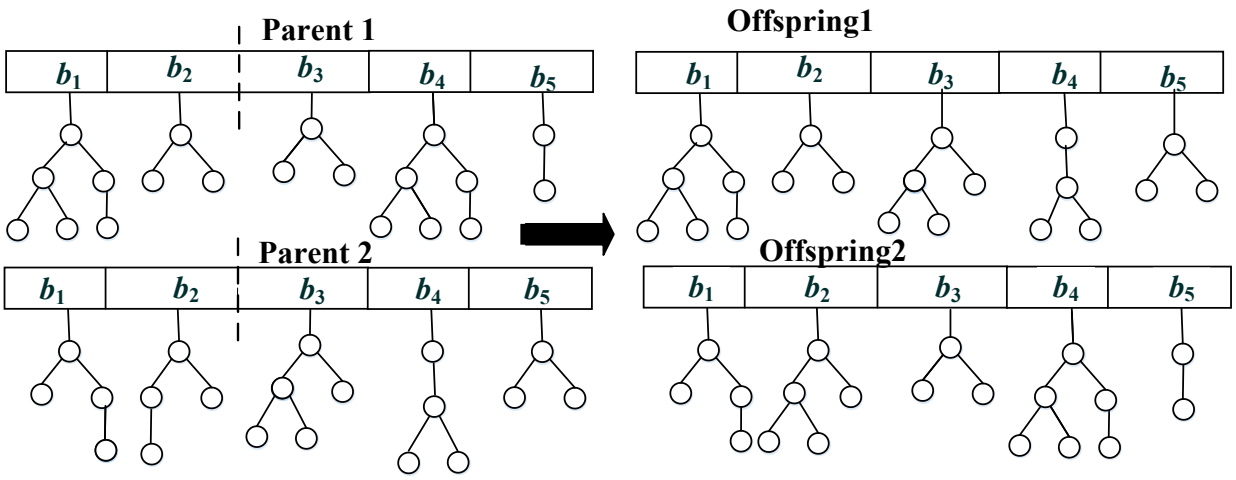
Figure 4. 3. Quantification of the fuzzy model's accuracy at the internal processing level [6].

4.3.4. Genetic operators

Genetic operators are used in GP to guide the algorithm towards a solution for a given problem. The GP uses three genetic operators to construct new programs (fuzzy models): two different crossover operators and one mutation. Crossover can be applied at two levels: (i) high level and (ii) low level. The low level is the space for manipulating the terminals and functions, whereas the high level is the space where the expressions can be exchanged. One-point high-level crossover is performed as illustrated by the following example. Suppose the first and second parent consists of four trees, say (G_1, G_2, G_3, G_4) and (g_1, g_2, g_3, g_4), respectively. First, a crossover point is randomly selected, e.g., the third three in this example. Thereafter, the trees beyond the crossover point in two parents are exchanged, and the following two new individuals are resulted: (G_1, G_2, g_3, g_4) and (g_1, g_2, G_3, G_4). However, in the case of low-level crossover, a tree is chosen at random from each parent. Then, the standard sub-tree crossover is applied, and the subtrees replace the parent trees; otherwise, the parents are transferred as unaltered individuals in the next generation. Figure

4.4 presents an example of the crossover operators on the multi-tree individuals with five expressions (trees). The dashed lines indicate the subtrees that have been exchanged between two parents.

(a) High-level crossover



(b) Low-level crossover

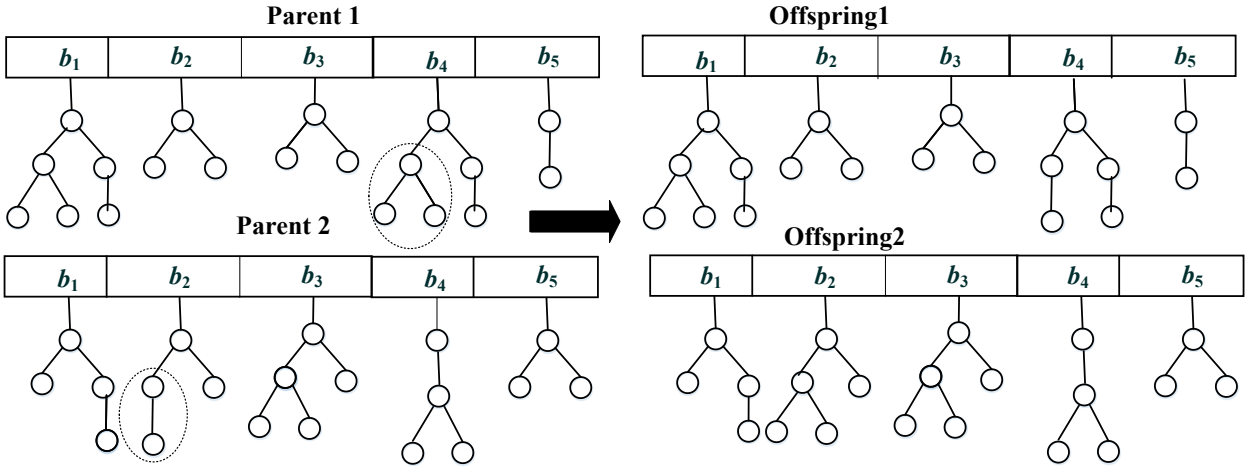


Figure 4. 4. Examples of crossover operators, (a) high-level and (b) low-level crossover.

The point mutation is performed as follows: a randomly selected node in the offspring is replaced with another random primitive (terminal or function) of the same arity; that is, a randomly selected terminal (say fuzzy set) is substituted with another terminal, and a randomly selected operator is replaced with an operator from the function set. Figure 4.5 visualizes an example of the mutation operator, in which fuzzy set a_1 and fuzzy operator t are mutated randomly to fuzzy set a_3 and operator s, respectively. Intuitively, high-level crossover affects the output more profoundly than low-level crossover or mutation operator.

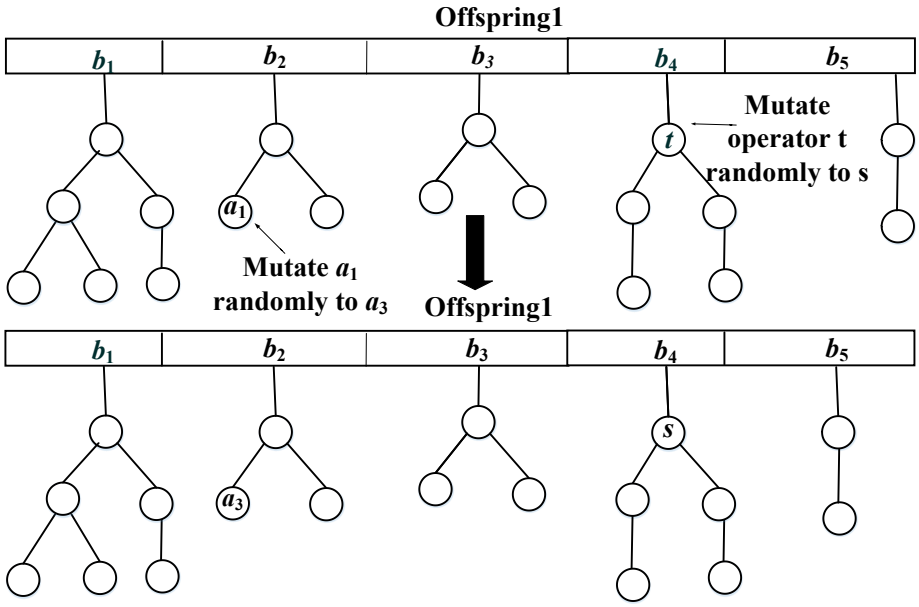


Figure 4. 5. An example of the point mutation operator.

4.3.5. Parametric optimization of the fuzzy model with gradient descent learning

Suppose $\boldsymbol{b}_k \in [0,1]^{c_2}$ is the vector of membership achieved by the logic processing core (GP) in the previous step. It is transformed as $z_k = f(W \cdot \boldsymbol{b}_k)$ where W is a $c_2 \times c_2$ weight matrix, and f is a nonlinear sigmoidal function. Then the output of the model is calculated by the center of gravity

mechanism, so $y_i = \boldsymbol{z}_i^{\mathrm{T}} . V$, where $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{c_2}]$ are the representatives of the output space which are achieved by FCM in the first step of architecture explained in Section 4.2. A gradient-based method of learning is desirable to optimize the entries of W. Following the notation presented in Section 4.2, the learning process can be described as follows:

For a given set of N training data $(\boldsymbol{x}_k, target_k)$, k=1, 2, …, N, adjust the network's connections to minimize the performance index of the model (Q). Here performance index is expressed at the external (numeric) level of processing as visualized in Figure 4. 6. As we are working on the regression problems in this study, Root Mean Square Error (RMSE) is used as the performance index (Q) which is defined as follows:

$$Q = \sqrt{\frac{1}{N}\sum_{k=1}^{N} (y_k - target_k)^2} \tag{4.2}$$

where y_k is the output of the model for the given input $\boldsymbol{x}_k$. The network's connections are updated as follows:

$$W(iter+1) = W(iter) - \alpha \frac{\partial Q}{\partial W} \tag{4.3}$$

where $W(iter)$ is the network's connections in the $iter^{\text{th}}$ iteration of the algorithm.

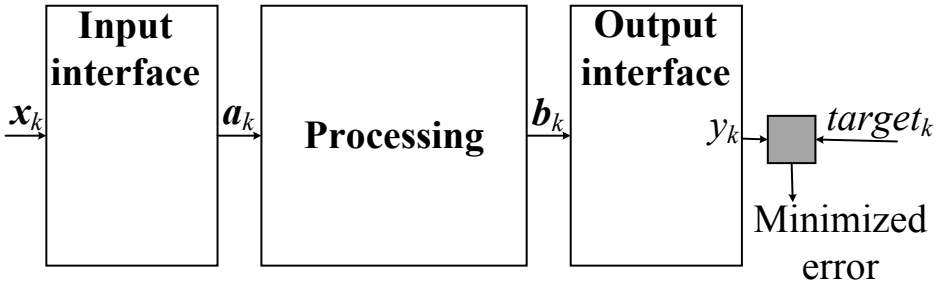


Figure 4. 6. Quantification of the fuzzy model's accuracy at the external processing level [6].

4.4. Experimental results

Numerical instances for the performance evaluation of the proposed approach are provided here. Synthetic data and two high-dimensional regression datasets come from the Penn Machine Learning Benchmarks (https://github.com/EpistasisLab/penn-ml-benchmarks). Eight benchmarks are also taken from the ICOS PSP Benchmarks Repository (http://ico2s.org/datasets/psp_benchmark.html). The experiments are conducted on a 1.30 GHz core i7 PC with 16 GB of RAM under the Matlab environment.

4.4.1. Synthetic data

The following experiment illustrates adjusting the GP for the automated design of logic function by a 3-D synthetic data. A 1000 3-D input dataset is generated over $[0, 1]^2$ using a uniform distribution. Then, the outputs of the dataset are formed by the logic expression which is shown in Figure 4.7. Next, the GP is employed to form a combinational logic expression. Function and terminal sets are composed of fuzzy operators {t, s, Negation, Xor, Eq, Imp, Nand, Nor}, and {x_1, x_2, x_3}, respectively. In this implementation, min and max are respectively used for t-norm (t) and t-conorm(s). Note that because of the nature of data that is already in the range [0, 1], encoding (fuzzification) and decoding (defuzzification) are not required in this example; thus, the GP is employed for the numeric values rather than the membership grades. In the experiments, we use 10-fold cross-validation. The parameters of GP after a fine-tuning have been chosen, as illustrated in Table 4.1. Standard sub-tree crossover and point mutation [52] are employed in this experiment.

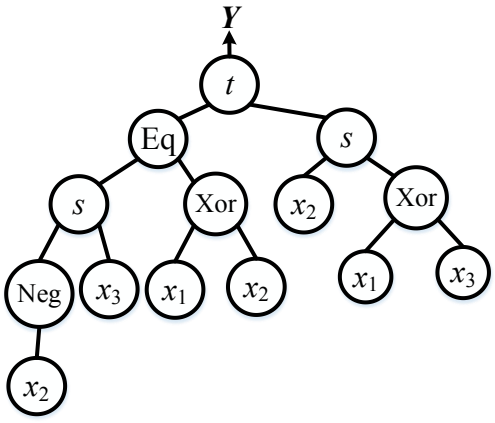


Figure 4. 7. The logic expression shown in the form of the outputs of the synthetic dataset.

Table 4. 1. Parameters of the GP

Parameter	Range	Selected value
Population size	[15-20]	20
Depth of solution in the initial generation	[3-6]	5
Crossover probability	[0.5-1]	0.8
Permutation probability (per node)	[0.05-0.2]	0.1
No. of generations	[30-60]	40
Tournament size	[2-5]	5
Population size	[15-20]	20

Figure 4.8 shows the values of performance indices resulting over the different generations of the GP. This figure indicates the evolutionary process performed by the GP over all generations.

The best logic expression achieved by the GP is visualized in Figure 4.9. The values of RMSE for training and testing data formed by this expression are 1.651E-17 and 3.73E-17, respectively. Although the expression formed by the GP seems different from the expression shown in Figure 4.7, the RMSE values achieved by the GP are very small for both the training and testing data.

These small values notably indicate the equivalence of the expression generated by the GP and the original expression formed in Figure 4.7.

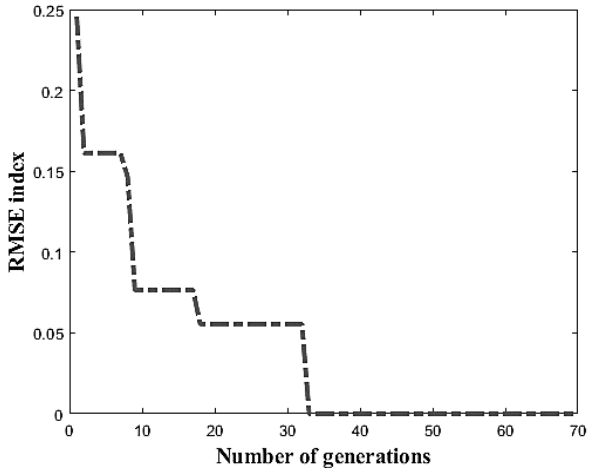


Figure 4. 8. The values of performance index for the training data over the generations.

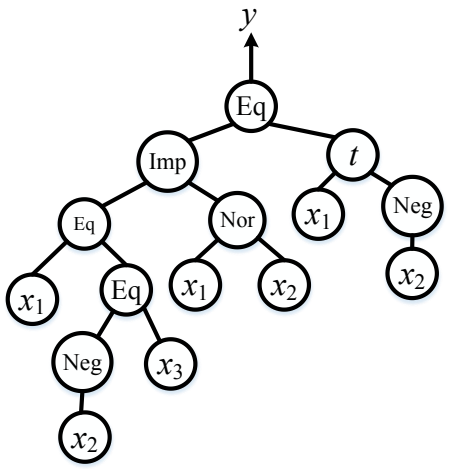


Figure 4. 9. The logic expression formed by the GP for multi-inputs single output data.

4.4.2. Synthetic high-dimensional datasets

As mentioned previously, one of the issues in the design of high-dimensional TSK models is the distance concentration problem. Two datasets (generated from the Friedman function) have been taken from the Penn Machine Learning Benchmarks to observe this issue.

The first data set is fri_c4_1000_100 composed of 1,000 instances with 100 features. We perform FCM on the input features with c=3 (the number of clusters) and m=2 (fuzzification coefficient) and portray distance among the clusters' prototypes in Figure 4.10. This figure illustrates how the increase of dimensionality impacts the distances among the prototypes. The x-axis represents dimensionality, and the y-axis visualizes the average distance among the prototypes. The destructive effect of high-dimensionality to the FCM is noticeable as the prototypes tend to position close to each other with the increase of dimensionality; that is, the distance among them goes to zero with increasing dimensionality of space; thus, the membership values would be uniformly distributed among all clusters.

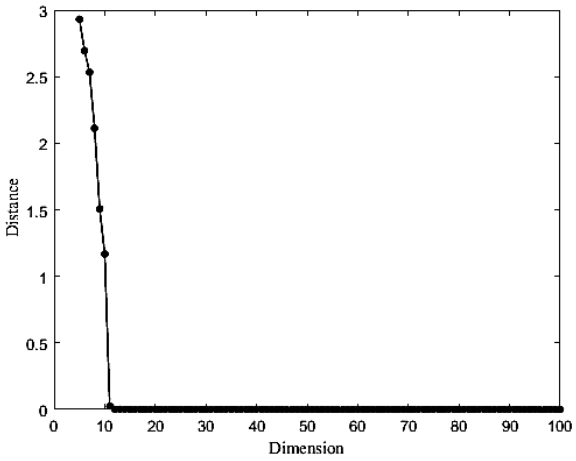
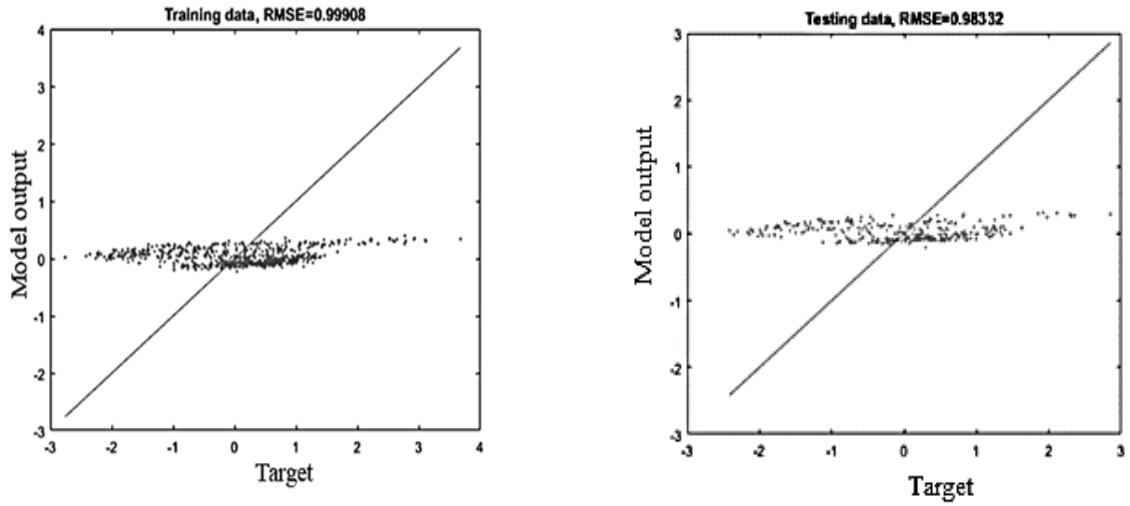


Figure 4. 10. Impact of high-dimensionality on the average distance among the prototypes in fri_c4_1000_100 dataset.

Generally, fuzzy sets in the antecedent of a standard TSK model are built using FCM. Thus, the failure of FCM to find suitable clusters (fuzzy sets) in high-dimensional data will intuitively have a negative impact on the performance of the TSK model. To observe this issue, we form a zero-order TSK model directly from training data in fri_c4_1000_100; the rule's antecedents in the TSK are built using FCM with c=3, m=2, and the consequences are constants estimated by

minimizing LSE criterion [6]. 70% of the data is selected randomly as training data to construct the model, and the remaining data are used for testing purposes. Figure 4.11 visualizes the performance of the TSK model on the training and testing data. In this figure, the *x*-axis and *y*-axis represent the target and model output values, respectively. It is shown that the model has poor performance for both training and testing cases; that is, the output is relatively constant and mostly far from the target. This limitation is caused by the distance concentration problem and its negative influence on FCM and, consequently, the TSK.



(a) (b)

Figure 4. 11. Performance of the TSK model on the fri_c4_1000_100 dataset, (a) training data, and (b) testing data

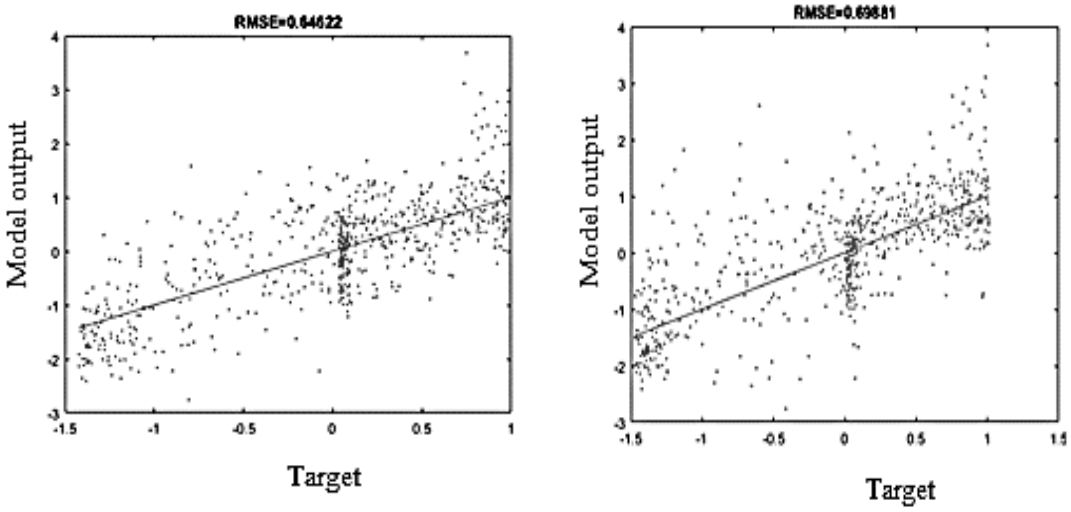
In the next experiment, FCM is applied independently to each variable to partition both input and output variables into three clusters, i.e., c_1=3, c_2=3. Then, the GP-based fuzzy model is formed after fine-tuning using the parameters given in Table 4.2.

Figure 4.12 illustrates the performance of the GP-based fuzzy model with respect to the training and testing data. The results show that using FCM on the individual variables improves the results

compared to the standard TSK. The performance index of the optimal structure obtained by the GP-based model for the training data is 0.64 as opposed to 0.99 for the TSK model and 0.69 versus 0.98 for the testing data. This is reasonable because we have eliminated the issue of distance concentration in the proposed GP-based model; that is, we now are dealing with single features rather than whole high-dimensional space.

Table 4. 2. Parameters of the GP and gradient descent

Parameter	Range	Selected value
Population size	{500,1k,2k}	2k
Depth of solution in the initial generation	[3-6]	6
Crossover probability	[0.5-1]	0.85
Permutation probability (per node)	[0.05-0.2]	0.2
No. of generations	[30-70]	50
Tournament size	[2-5]	5
Learning rate	{0.01, 0.1, 0.2}	0.01
No. of iterations of gradient descent	{30k, 40k, 50k}	50k
m: fuzzification coefficient	{1.1,1.5,2,2.5,3}	2



(a) (b)

Figure 4. 12. Performance of the GP-based fuzzy model on the fri_c4_1000_100 dataset, (a) training data, and (b) testing data

The optimal structure of logic expressions derived through structural and parametric optimization can be interpreted in the form of Figure 4.13. As previously stated, the number of fuzzy sets for each individual input and output variable is set to 3, say S (Small), M (Medium), and L (Large). These expressions can also be represented in the form of fuzzy rules.

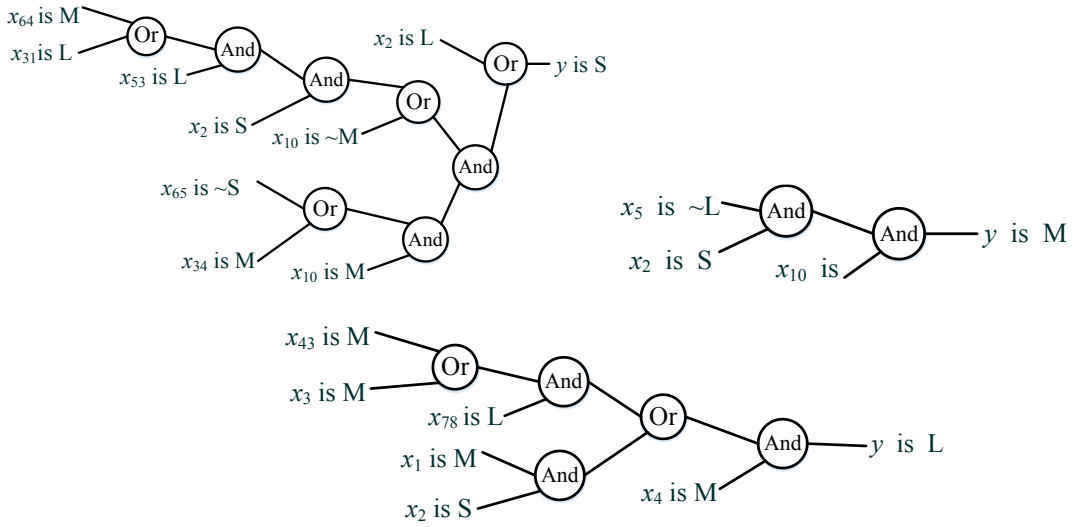
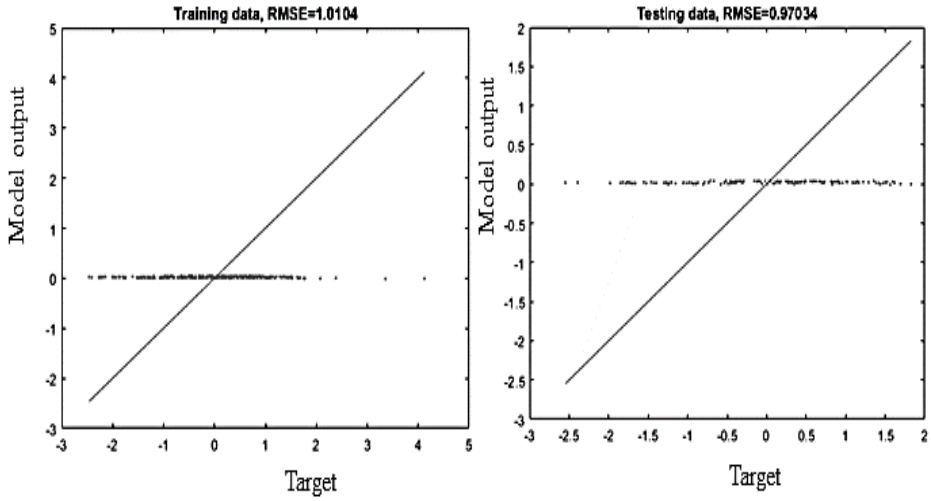


Figure 4. 13. Expressions derived through the GP-based model for the fri_c4_1000_100 data

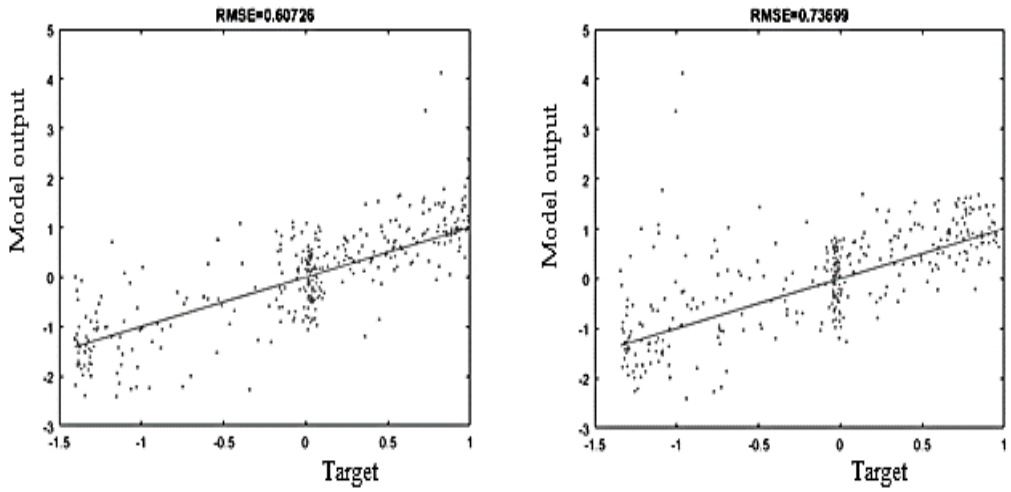
Note that the optimal structure contains only some of the input variables from the fri_c4_1000_100 data, not all of them. In other words, the GP can select some variables that are viewed as essential, performing implicit feature selection guided by the RMSE criterion. Also, the trees' depths are limited to a maximum depth of 6, which is set in the initial configuration.

The following dataset is fri_c4_500_100, composed of 500 instances with 100 features. The same experiments as those carried out on fri_c4_1000_100 are performed on this dataset, and comparison results are visualized in Figure 4. 14. This figure illustrates a similar tendency to the previous dataset. That is, in this high-dimensional dataset, the TSK shows poor performance

compared to the GP-based fuzzy model. In other words, the RMSE error for the GP model is far smaller than the TSK model for the training data (0.60 versus 1.01) and for the testing data (0.97 versus 0.73). These two experiments indicate the proposed architecture's suitability to form fuzzy models built for handling high-dimensional data.



(a) The TSK model



(b) The GP-based model

Figure 4. 14. Comparison results for the fri_c4_500_100 dataset, (a) the TSK model, and (b) the GP-based model

4.4.3. Real-world high-dimensional datasets

We use eight real-world high-dimensional datasets (Window2 to Window9) that are available in the ICOS PSP Benchmarks Repository. These datasets are derived from the same problem, Protein Structure Prediction (PSP), which predicts the 3-D structure of a protein relying on amino-acid structural variables. These datasets differ from each other based on the number of input variables, varying from 100 (Window2) to 380 (Window9), increasing in steps of 40. There are 257,560 instances in all the datasets.

The proposed GP-based fuzzy model, which employs augmented logic operators such as Xor, Nor, Nand, Equivalence as well as basic fuzzy operators, say *t*-norms, *t*-conorms, and negation, is compared with two other models: (i) the GP model utilized only basic fuzzy operators, namely *t*-norm, *t*-conorm, and negation, and (ii) zero-order TSK. The results are recorded in Table III. The number of clusters is set to c_1=3 and the number of clusters in output spaces (c_2) is varied between 3, 5 and 7. Product and probabilistic sum are used as the *t*-norm and *t*-conorm operators, respectively. Table 4.3 shows that using FCM on the individual variables in the GP models improves the results compared to the standard TSK; this table indicates the dominance of the proposed GP fuzzy model (with augmented logic operators) over the standard TSK and the GP model (with only basic logic operators) regarding the RMSE performance index; however, the running time of GP methods are naturally far higher than the TSK. Also, Table 4.3 intuitively illustrates that increasing the number of clusters in the output space leads to the improvements of performance indices in both models (mainly in the training data).

The optimal structure of logic expressions achieved by the proposed GP-based fuzzy model is visualized in Figure 4.15. This model is built based on the Window2 dataset and c_1=3, c_2=3. Still, it is noticeable that the GP can select only some of the variables which are viewed as essential.

Table 4. 3. The comparative analysis for the real-world datasets

Algorithm	No. of clusters		TSK			Proposed GP(with augmented logic operators)			Proposed GP(with basic logic operators)		
Dataset	C_1	C_2	RMSE (train)	RMSE (test)	Time (s)	RMSE (train)	RMSE (test)	Time (s)	RMSE (train)	RMSE (test)	Time (s)
Window2	3	3	5.20	5.23	279.84	**4.70**	**4.66**	2540.14	4.80	4.93	2200.14
Window3	3	3	6.23	5.76	280.25	**5.12**	**4.98**	3089.12	5.21	5.02	2870.66
Window4	3	3	5.68	5.77	292.57	**4.61**	**5.07**	2670.33	4.68	5.11	2600.06
Window5	3	3	5.82	5.70	310.12	**4.54**	**4.81**	3560.13	4.60	4.86	3331.12
Window6	3	3	5.51	5.63	300.17	4.94	5.11	2960.14	**4.90**	**5.08**	2900.18
Window7	3	3	5.93	5.87	320.14	**4.53**	**4.59**	3180.12	4.55	4.66	3050.43
Window8	3	3	5.44	5.56	300.16	**5.16**	**5.23**	2870.77	5.22	5.43	2840.15
Window9	3	3	6.19	6.78	353.15	**5.50**	**5.48**	3660.07	5.66	5.52	3600.71
Window2	3	5	5.15	5.20	469.41	**4.72**	**4.77**	3300.14	4.75	4.83	3050.71
Window3	3	5	6.09	6.23	517.82	**4.94**	**4.87**	3240.05	5.10	5.02	2980.14
Window4	3	5	5.53	5.63	532.13	**4.53**	**4.64**	3129.91	4.58	4.78	3030.63
Window5	3	5	5.93	5.87	555.76	**4.53**	**4.58**	3643.61	4.55	4.63	3345.05
Window6	3	5	5.43	5.51	593.14	**4.81**	**4.91**	3730.14	4.84	4.96	3350.19
Window7	3	5	5.97	6.03	600.93	4.43	4.50	3757.91	**4.35**	**4.41**	3300.93
Window8	3	5	5.31	5.49	612.14	**5.15**	**5.21**	3960.17	5.20	5.42	3512.23
Window9	3	5	6.05	6.45	503.29	**5.38**	**5.37**	3800.18	5.50	5.64	3700.18
Window2	3	7	5.08	5.10	921.18	**4.65**	4.63	3618.13	4.75	4.78	3440.37
Window3	3	7	5.96	5.74	806.21	**4.90**	4.93	3630.16	4.97	4.95	3100.05
Window4	3	7	5.60	5.68	879.11	**4.45**	4.55	3790.46	4.58	4.60	3390.22
Window5	3	7	5.77	5.78	913.13	**4.49**	4.55	3903.82	4.53	4.67	3601.10
Window6	3	7	5.38	5.70	950.14	**4.73**	4.81	4009.54	4.80	4.89	3367.17
Window7	3	7	5.76	5.85	946.66	4.51	4.66	3780.17	**4.44**	**4.50**	3600.16
Window8	3	7	5.25	5.57	1020.11	**5.02**	5.13	3876.23	5.22	5.36	3612.18
Window9	3	7	5.93	6.21	1003.13	**5.29**	5.27	3954.03	5.37	5.57	3800.91

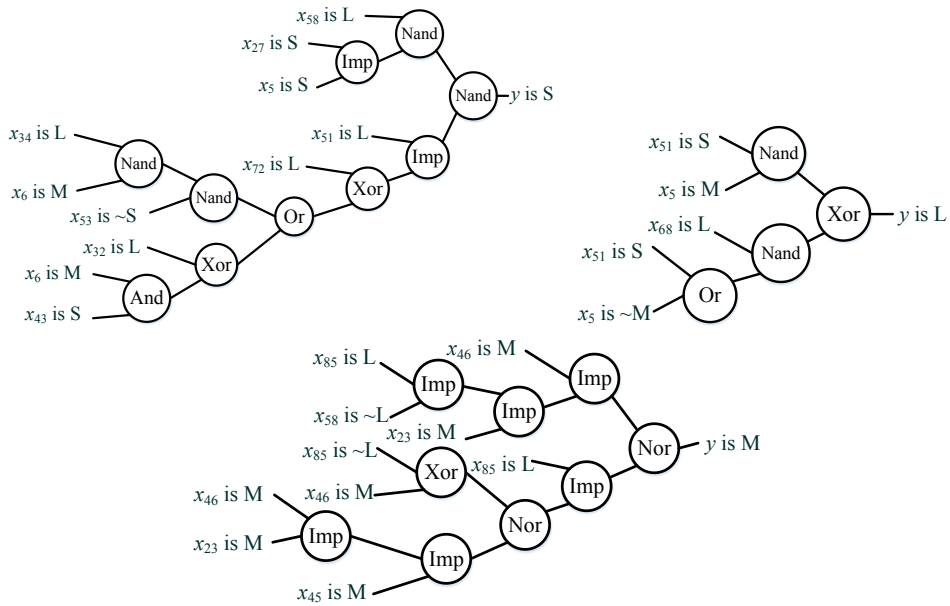


Figure 4. 15. Expressions obtained through the GP-based model for the Window2 dataset

In what follows, the influence of different *t*-norm and *t*-conorm operators on the performance of the fuzzy model formed by the GP is examined. Some widely used *t*-norms (minimum, product, Lukasiewicz and drastic product) and *t*-conorm (maximum, probabilistic sum, Lukasiewicz and drastic sum) are employed in this study. See Table 4.4.

Table 4. 4. Selected examples t-norms and t-conorms

t-norms		*t*-conorms	
Minimum	$a\ t_m\ b = \min(a,b)$	Maximum	$a\ s_m\ b = \max(a,b)$
Product	$a\ t_p\ b = ab$	Probabilistic sum	$a\ s_p\ b = a+b-ab$
Lukasiewicz	$a\ t_l\ b = \max(a+b-1,0)$	Lukasiewicz	$a\ s_l\ b = \min(a+b,1)$
Drastic product	$a\ t_d\ b = \begin{cases} a, \text{ if } b=1 \\ b, \text{ if } a=1 \\ 0, \text{ otherwise} \end{cases}$	Drastic sum	$a\ s_d\ b = \begin{cases} a, \text{ if } b=0 \\ b, \text{ if } a=0 \\ 0, \text{ otherwise} \end{cases}$

We present the results for the commonly used *t*-norms and *t*-conorms in Table 4.5. The table illustrates that, for the Window 3 dataset, using the product and the probabilistic sums as the representative of *t*-norm and *t*-conform in the design of the GP-based fuzzy model leads to improved performance compared to the other dual norms.

Table 4. 5. The performance of model using *t*-norms and *t*-conorms

t-norm	*t*-conorm	*Q* (training)	*Q* (testing)
Minimum	Maximum	5.33	5.14
Product	Probabilistic sum	**5.12**	**4.98**
Lukasiewicz	Lukasiewicz	5.14	5.03
Drastic product	Drastic sum	5.35	5.42

The logic expression obtained by the product and the probabilistic sum as *t*-norm and *t*-conorm, respectively, is visualized in Figure 4.16. It is seen that the structure of the fuzzy model involves logic operators such as Xor, Nor, Nand, Equivalence, and Implication as well as basic operators. Also, as the initial population is generated by grow method, the evolved individuals are less likely to be full trees. In other words, nodes at the intermediate levels are taken from both function and terminal sets, but after the depth limit, say maximum 6, is met, the leaves are chosen from terminals.

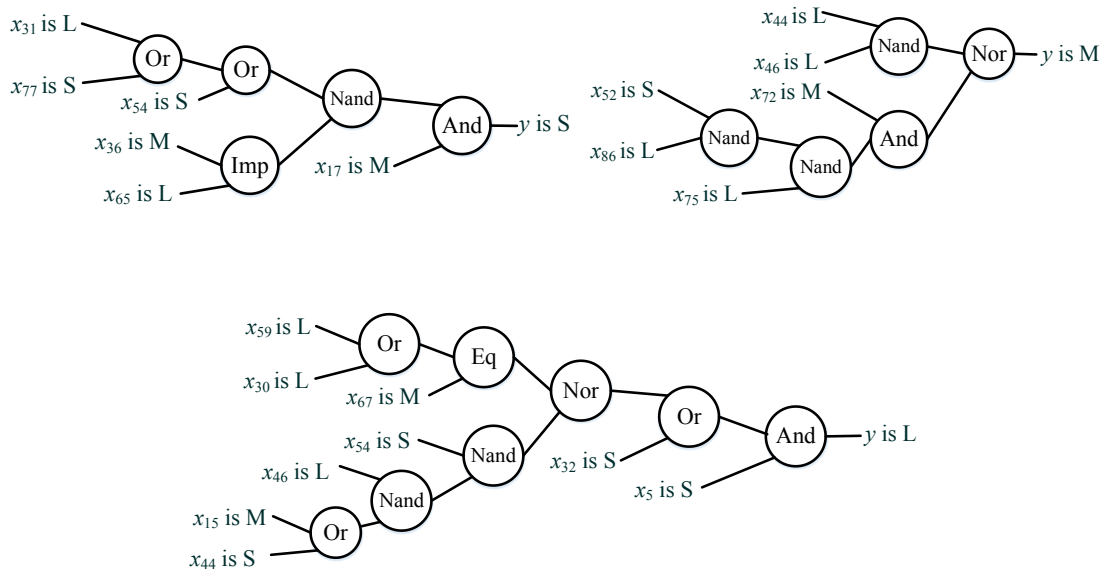


Figure 4. 16. Expressions obtained through the GP-based model for the Window3 dataset

To sum up, four major aspects have been explored in this experimental study.

(1) The distance concentration issue in high-dimensional data and, consequently, its destructive effect on the TSK model's performance has been examined.

(2) A new architecture based on integrating the FCM, the GP, and gradient descent has been proposed. This model comes with the flexibility of a logic-oriented structure of fuzzy models and is carried out for high-dimensional data alleviating the detrimental effect of distance concentration.

(3) The proposed model is compared with two other models: (i) the GP model utilized only basic fuzzy operators, namely *t*-norm, *t*-conorm, and negation, and (ii) zero-order TSK. The experimental results illustrate the proposed model's superiority (in terms of accuracy) over the other models in benchmark problems.

(4) Several combinations of *t*-norms and *t*-conorms have been employed in the evolutionary process, and the influence of different operators on model performance has been examined.

4.5. Conclusions

In this chapter we proposed a GP-based method for the automatic extraction of the fuzzy models from input-output data. In the proposed method, the structure of a fuzzy model is represented by a multi-tree individual, and GP is utilized to find the optimal fuzzy model. The idea of admitting augmented fuzzy operators (e.g., Xor, Nor, Nand, Equivalence, and Implication) in addition to basic operators (viz., *t*-norm, *t*-conorm, and negation) is also introduced, and then the effects of using different *t*-norms and *t*-conorms on the model's performance are discussed.

This chapter also addresses the distance concentration issue in the design of high-dimensional fuzzy models, where the increase of dimensionality may converge all the pairwise distances (dissimilarities) to the same value, leading to failure of the fuzzy model. In this chapter, FCM is applied independently to each variable, rather than to all variables (dimensions) at once, to eliminate the problem of distance concentration. Subsequently, the model's structural optimization is accomplished using multi-tree genetic programming, and parametric optimization is performed via gradient-based learning.

We applied our GP-based fuzzy model to synthetic datasets, specifically, some regression datasets from the Penn Machine Learning Benchmarks and the ICOS PSP Benchmarks Repository,

and we compared our methodology with other approaches. In the experiments, our GP-based model (with the augmented logic operators) attained higher performance compared to the GP model with the basic logic operators and the standard TSK fuzzy model available in the literature.

Chapter 5 A PSO-aided Development of TSK Fuzzy Models Employing Various Non-linear Consequences

This chapter's primary aim is to generalize Takagi–Sugeno–Kang (TSK) fuzzy rule-based systems (FRBS) by involving non-linear functions in the rules' consequences. In the current literature, most of the TSK models with polynomial (linear) consequences have been studied; however, the TSK models' design with non-linear consequences has not been discussed to a great extent. This chapter's originality comes with the generalization of the TSK model, which employs a family of non-linear and linear local models rather than only linear models forming the rules' consequences. The proposed modification reduces the model's complexity (number of rules) while preserving the desired accuracy.

In this chapter, we develop an architecture in which information granules (fuzzy sets) in the antecedents of the rules are created by the FCM clustering algorithm. The structure of non-linear consequences is also identified by Particle Swarm Optimization (PSO) and the parameters of consequences are realized by minimizing the LSE optimization criterion. A number of experimental studies, along with the comparative analysis including synthetic and publicly available datasets, are provided to demonstrate the effectiveness of the proposed approach.

5.1. Non-linear TSK models

The functional fuzzy models (TSK) are utilized to describe complex non-linear systems by a collection of rules whose consequences are local functions. These functions could be linear, non-

linear, differential equations, or neural networks [6]. Although the model with linear consequences possesses the universal approximation property, but in practice, the number of rules to achieve the desired accuracy is high. In the current literature, most of the TSK models with polynomial (linear) consequences have been studied; however, the TSK models' design with non-linear consequences has not been discussed notably. This study's motivation is concerned with the development of data-driven FRBS by involving a family of non-linear and linear local models rather than only linear models. This modification reduces the model complexity (number of rules) while preserving the model's desired accuracy.

The idea of using non-linear functions in the consequences of fuzzy rules was already discussed in the literature, but they are limited only to particular control systems [58-61]. In this context, non-linear TSK has been employed to avoid using an excessive number of rules to approximate the system dynamics. Two methods come up: (i) the first one involves using TSK models with polynomial consequence [54-55], and (ii) the other one is the method introduced in the literature [56-57], in which linear parts plus a sector-bounded non-linearity are considered to generate non-linear local models.

Despite these works, to the best of our knowledge, there are no studies so far addressing the structural and parametric optimization of fuzzy rules with non-linear consequences; in other words, an allocation of various non-linear functions for the consequences of rules has not been discussed enough. In this study's proposed architecture, PSO is utilized to obtain the optimal arrangement of non-linear functions standing in the consequences of rules whereas the parameters of consequences are estimated by the minimizing LSE optimization criterion. In this development, the FCM clustering algorithm is employed to develop fuzzy sets in the antecedents of rules.

5.2. Functional fuzzy rules

As we already discussed in chapter 2, a well-studied form of FRBS is a TSK model. The design of the TSK fuzzy model provides a systematic approach for generating fuzzy rules from a given input-output data set. The following expression is the common form of TSK style rules:

$$\text{IF } \boldsymbol{x} \text{ is } A_i \text{ THEN } y \text{ is } f_i(\boldsymbol{x},\boldsymbol{a_i}) \tag{5.1}$$

where i = 1, 2,…, c, and c is the number of rules, $\boldsymbol{x}$ is a n-dimensional input variable, A_i is the membership function of i^{th} fuzzy set in the input space, y is the predicted output by the numeric function $f_i(\boldsymbol{x},\boldsymbol{a_i})$. The function f_i can take any format, say, linear, non-linear, differential equations, or neural networks. Several types of consequences are used in our study.

$$\text{Type 1(constant): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0} \tag{5.2}$$

$$\text{Type 2 (linear): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0}+ \boldsymbol{a_{i1}}^{\mathrm{T}}\boldsymbol{x} \tag{5.3}$$

$$\text{Type 3 (quadratic): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0}+ \boldsymbol{a_{i1}}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{a_{i2}}^{\mathrm{T}}\boldsymbol{x}^2 \tag{5.4}$$

$$\text{Type 4 (exponential): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0} + a_{i1} \exp(\boldsymbol{x}) \tag{5.5}$$

$$\text{Type 5 (square root): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0} + a_{i1} \operatorname{sqrt}(\boldsymbol{x}) \tag{5.6}$$

$$\text{Type 6 (sine): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0} + a_{i1} \sin(\boldsymbol{x}) + a_{i2} \sin(2\boldsymbol{x}) + \ldots + a_{it} \sin(t\boldsymbol{x}) \tag{5.7}$$

$$\text{Type 7 (cosine): } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0} + a_{i1} \cos(\boldsymbol{x}) + a_{i2} \cos(2\boldsymbol{x}) + \ldots + a_{it} \cos(t\boldsymbol{x}) \tag{5.8}$$

where the vector $\boldsymbol{a_{ij}}$ and singleton a_{ij} represents j^{th} vector of coefficients and a scaler coefficient in the i^{th} rule, respectively. $\boldsymbol{x}^2$ denotes an element-wise multiplication of vector $\boldsymbol{x}$. The variable t is the number of terms (harmonics) extended by the consequences types 6 and 7. These functions are similar to how Fourier transform decomposes signals onto a bank of sine and cosine functions.

By forming a number of fuzzy rules and combining them as a set of "if-then", we build a fuzzy rule-based model. When arranging all the rules together involving their antecedents, the output of the model, say $\hat{y}$, is aggregated by taking the weighted average of the output of each rule as follows,

$$\hat{y}=\sum_{i=1}^{c} A_i(\boldsymbol{x})\, f_i(\boldsymbol{x},\boldsymbol{a_i}) \tag{5.9}$$

To form the fuzzy sets, A_i which define the antecedents of the fuzzy rules, a common strategy and the one used in this study is FCM clustering, which takes the fuzzy partitions defined by cluster prototypes and the partition matrix U. In the standard TSK fuzzy model, $f_i(\boldsymbol{x},\boldsymbol{a}_i)$ in the consequences of the rules is adopted as a linear function, and in some cases, the function is simplified as a constant value. However, the consequence could also be any non-linear function. Suppose that the consequences of rules are a finite combination of some non-linear sine functions.

$$R_i: \text{IF } \boldsymbol{x} \text{ is } A_i, \text{ THEN } f_i(\boldsymbol{x},\boldsymbol{a_i})=a_{i0}+a_{i1}\sin(\boldsymbol{x})+a_{i2}\sin(2\,\boldsymbol{x})+\ldots+a_{it}\sin(t\,\boldsymbol{x}) \tag{5.10}$$

where local models are non-linear regarding input-output mapping; however, they are still linear models of parameters, so the parameters of local functions can be identified by using the LSE performance index. Let us see the process of determining the parameters in detail. The parameters of a local function can be shown in the form of $a_i = [a_{i0}, a_{i1}, \ldots, a_{it}]$. The output of the model is expressed in the following form:

$$\hat{y}=\sum_{i=1}^{c} A_i(\boldsymbol{x})\,.\boldsymbol{a_i}^{\mathrm{T}}\begin{bmatrix}1\\ \sin(\boldsymbol{x})\\ \sin(2.\,\boldsymbol{x})\\ \cdots\\ \sin(t.\,\boldsymbol{x})\end{bmatrix}=\sum_{i=1}^{c}\boldsymbol{a_i}^{\mathrm{T}}\begin{bmatrix}A_i(\boldsymbol{x})\\ A_i(\boldsymbol{x})\sin(\boldsymbol{x})\\ A_i(\boldsymbol{x})\sin(2.\,\boldsymbol{x})\\ \cdots\\ A_i(\boldsymbol{x})\sin(t\,.\,\boldsymbol{x})\end{bmatrix} \tag{5.11}$$

Suppose

$$z_i(\boldsymbol{x})=\begin{bmatrix} A_i(\boldsymbol{x}) \\ A_i(\boldsymbol{x})\sin(\boldsymbol{x}) \\ A_i(\boldsymbol{x})\sin(2.\boldsymbol{x}) \\ \cdots \\ A_i(\boldsymbol{x})\sin(t\,.\,\boldsymbol{x}) \end{bmatrix} \tag{5.12}$$

so

$$\hat{y}=\sum_{i=1}^{c} \boldsymbol{a_i}^T z_i(\boldsymbol{x})=\sum_{i=1}^{c} z_i^{\mathrm{T}}(\boldsymbol{x})\,\boldsymbol{a}_i \tag{5.13}$$

Let us use the following vector notation to arrange all parameters of the models.

$$\boldsymbol{a}=\begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \\ \cdots \\ \boldsymbol{a}_c \end{bmatrix} \tag{5.14}$$

and

$$f(\boldsymbol{x})=\begin{bmatrix} z_1(\boldsymbol{x}) \\ z_2(\boldsymbol{x}) \\ z_3(\boldsymbol{x}) \\ \cdots \\ z_c(\boldsymbol{x}) \end{bmatrix} \tag{5.15}$$

The collection of N-input-target data is then organized in the following matrix format:

$$\boldsymbol{y}=\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdots \\ y_N \end{bmatrix} \tag{5.16}$$

and

$$\boldsymbol{F}=\begin{bmatrix} \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}_1) \\ \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}_2) \\ \cdots \\ \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}_N) \end{bmatrix}=\begin{bmatrix} z_1{}^{\mathrm{T}}(\boldsymbol{x}_1) & z_2{}^{\mathrm{T}}(\boldsymbol{x}_1) & \ldots & z_c{}^{\mathrm{T}}(\boldsymbol{x}_1) \\ z_1{}^{\mathrm{T}}(\boldsymbol{x}_2) & z_2{}^{\mathrm{T}}(\boldsymbol{x}_2) & \ldots & z_c{}^{\mathrm{T}}(\boldsymbol{x}_2) \\ \cdots & \cdots & \cdots & \cdots \\ z_1{}^{\mathrm{T}}(\boldsymbol{x}_N) & z_2{}^{\mathrm{T}}(\boldsymbol{x}_N) & \ldots & z_c{}^{\mathrm{T}}(\boldsymbol{x}_N) \end{bmatrix} \tag{5.17}$$

so $\quad \hat{\boldsymbol{y}}=\boldsymbol{F}\boldsymbol{a}$ (5.18)

With the sum of squared error $Q=\|\boldsymbol{y}-\widehat{\boldsymbol{y}}\|^2=(\boldsymbol{Fa}-\boldsymbol{y})^{\mathrm{T}}(\boldsymbol{Fa}-\boldsymbol{y})$ and its minimization with respect to $\boldsymbol{a}$, the optimal estimated parameters are expressed in the following format:

$$\boldsymbol{a}_{\text{optimal}} = (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{y} \tag{5.19}$$

Another variation of the TSK model is involving the prototypes obtained from clustering to the local function. Namely, $f_i(\boldsymbol{x},\boldsymbol{a_i})$ stands for a local function interpreted as a hyperplane governed by the following expression,

$$f_i(\boldsymbol{x},\boldsymbol{a_i})=w_i+a_{i1}\sin(\boldsymbol{x}-\boldsymbol{v}_i)+a_{i2}\sin(2(\boldsymbol{x}-\boldsymbol{v}_i))+\ldots+a_{it}\sin(t(\boldsymbol{x}-\boldsymbol{v}_i)) \tag{5.20}$$

where $\boldsymbol{v}_i$ is a cluster (prototype) capturing the location of the rule in the input space $\boldsymbol{R}^n$ and w_i is the corresponding value in the output space. As already discussed, the parameters of local functions can be similarly estimated using the (LSE) method.

5.3. Architecture of proposed fuzzy model

We develop an integrated architecture based on information granulation (fuzzy clustering) and PSO algorithm. The overall architecture of this methodology is shown in Figure 5.1.In this architecture, four main modules are responsible for forming the fuzzy model directly from input/targets data. These components are as follows: (i) the FCM, which is responsible for forming the fuzzy sets in the antecedents of rules. (ii) The optimization core, the most crucial module of the proposed architecture, benefits from a PSO to form optimal non-linear or linear consequences of rules as the structure of the fuzzy model. This module also employs the LSE method to determine the parameters of consequences. (iii) Rules aggregation module forms the numeric output of the model using (5.9) by aggregating the inferred values from individual rules. (iv) The

evaluation module which is used to calculate the cost function, viz, the root mean squared error (RMSE) of the fuzzy model over the evolution process.

In the introduced architecture, the optimization core receives information granules formed by the clustering module (FCM) from the numeric inputs. This module's primary role is to optimize the fuzzy model's structure by allocating diverse linear and non-linear local functions to the consequences of rules from the given input data as well as optimizing the parameters of the model. This is accomplished by building a model using various linear and non-linear functions (constant, linear, quadratic, sine, cosine, exponential and square root). This problem is combinatorial; thus, using the evolutionary (or swarm) computation is a compelling alternative to solve it. We focus on employing PSO in this study because of its less significant computing overhead for optimizing model structure compared to the other evolutionary methods.

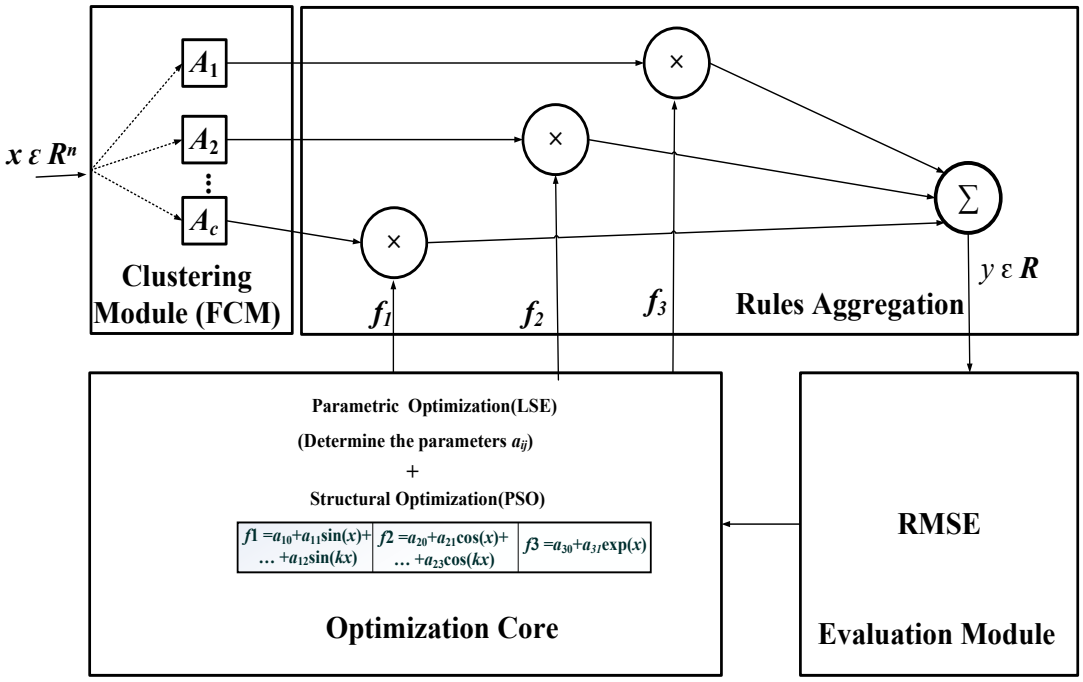


Figure 5. 1. Overall scheme of the proposed design methodology of the fuzzy model

In the proposed architecture, the allocation of diverse local functions to the TSK model's rules is represented by the particles. Each particle represents a vector of real values in the [0, 1) scale. The size of each particle (i.e., fuzzy model) is c, where c is the number of rules. As PSO is naturally working in real-valued representation, we need to decode the particle to an appropriate solution to calculate the particles' fitness over the evolution process. We consider several subintervals whose number depends on the number of functions as already discussed in Section 5.2. As seven various functions are used in this study, we arrive at the mapping of intervals to functions as follows: constant:[0,1/7), linear:[1/7,2/7], quadratic:[2/7,3/7), sine:[3/7,4/7), cosine:[4/7,5/7), exponential:[5/7,6/7] and square root[6/7,1). Suppose that we want to build a TSK model with four rules given seven functions that can be assigned as the consequences of rules. Assume that the position's content for the candidate solution (particle) is [0.78, 0.50, 0.70, 0.0.41], so for the given position the consequences of rules will be exponential, sine, cosine, and quadratic, respectively.

The objective in this study is forming a fuzzy rule-based model that involves the allocation of different linear and non-linear functions to the consequences of individual rules in such a way that the cost function of the model (e.g., the RMSE) becomes minimized. Assuming that $\hat{y}_k$ is the result produced by the fuzzy rule-based model for k^{th} input and y_k is k^{th} target output, the RMSE index Q is expressed as follows:

$$Q = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2} \tag{5.21}$$

5.4. Experimental studies

Numerical experiments completed for the performance evaluation of the proposed approach are provided here. Synthetic data and some publicly available regression datasets from different repositories are utilized in this study. In the experiments, we use a 5-fold cross validation method. The data set is divided into five subsets of equal size. For each case, a subset is selected as the test dataset and the remaining part of the data serves as the training dataset which is used to construct the model. The experiments are conducted on a 1.30 GHz core i7 PC with 16 GB of RAM under the Python environment.

5.4.1. Synthetic dataset

In this subsection, we demonstrate the proposed model's performance on two non-linear synthetic data sets which are defined in Table 5.1. We sample 600 data points from input space x in a uniform manner.

Table 5. 1. Synthetic datasets

Dataset	Function	Range
Synthetic dataset #1	$y=\frac{\sin(2x)}{x}$	$x \in [-4\pi, -4\pi]$
Synthetic dataset #2	$y = 0.2 \sin(2\pi x) + 0.2 x^3 + 0.3$	$x \in [0, 2]$

First, we develop the rule-based model, composed of c=7 rules, based on the synthetic dataset #1. The parameters of FCM and PSO after a fine-tuning have been chosen, as illustrated in Table 5.2. As already mentioned, seven various types of consequences form the search space of PSO in

our study. These functions are constant, linear, quadratic, exponential, square root, sine and cosine. In our experiments we consider only a finite family of harmonics, usually relatively small, so the variable *t,* which is the number of terms extended by the consequences of types 6 and 7, is set to 4.

Table 5. 2. Parameters of the model

Parameter	Range	Selected value
Swarm size	[30-50]	50
Number of iterations	[20-50]	30
inertia weight damping ratio	[0.6-0.95]	0.9
c_1	{0.3,0.4,0.5,0.6, 0.7}	0.5
c_2	{0.3,0.4,0.5,0.6, 0.7}	0.3
m:Fuzzification coefficient	{1.7, 1.9, 2, 2.2, 2.5}	2

In Figure 5.2, we visualize the values of performance index obtained in successive generations of the PSO. The model's evolution reaches a plateau after almost 30 iterations (early stopping), so we select 30 as the maximum number of iterations in this experiment.

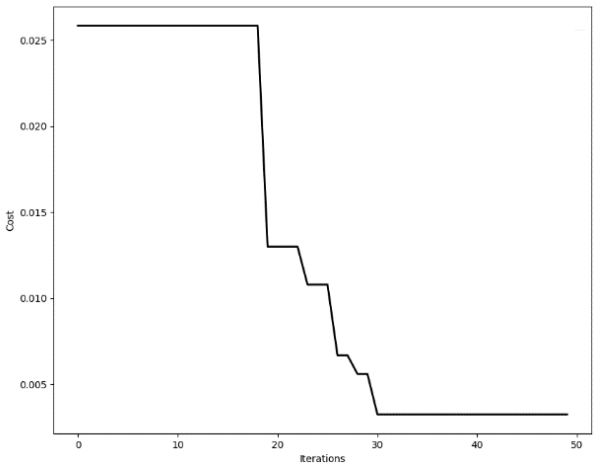


Figure 5. 2. The values of performance index over the generations

The allocation of consequences achieved by the PSO is also displayed in Figure 5.3(a). This figure visualizes the prototypes produced by the FCM and the outputs of each local model in the corresponding subspace. The allocated consequences for the rules are some sinusoidal and cosinusoidal functions among the seven types of linear and non-linear functions which we used in this study. These rules which are generated by the proposed architecture are as follows:

f_1= 0.01– 0.10 cos(x) + 0.01cos(2x)+ 0.04cos(3x) - 0.01cos(4x)

f_2= -0.04 - 0.05 sin(x) - 0.14 sin (2x) - 0.02 sin(3x) - 0.01 sin(4x)

f_3= 0.02 - 0.03 sin(x) - 0.33 sin (2x) - 0.04 sin(3x) + 0.01 sin(4x)

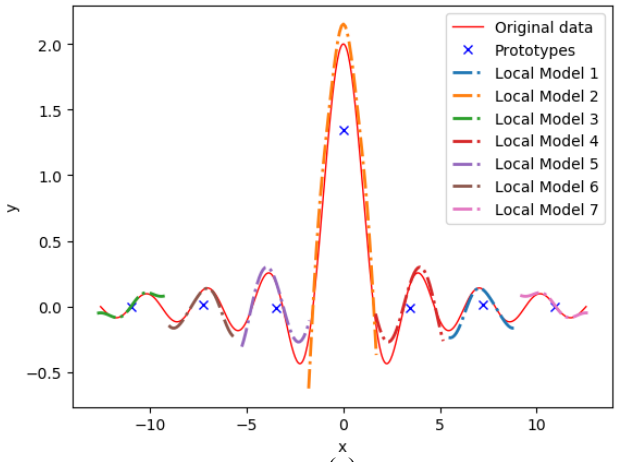
f_4= -0.05 + 2.14 cos(x) + 0.08 cos (2x) - 0.17 cos(3x) + 0.14 cos(4x)

f_5= 0.02 + 0.03 sin(x) + 0.33 sin (2x) + 0.04 sin(3x) + 0.01 sin(4x)

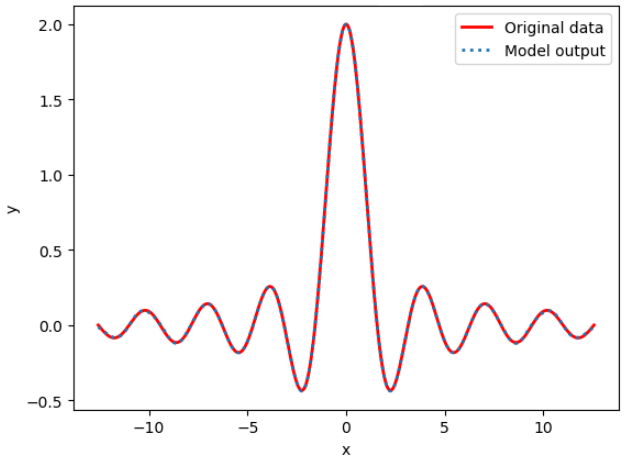
f_6= -0.05 + 0.05 sin(x) - 0.15 sin (2x) - 0.01 sin(3x) - 0.01 sin(4x)

f_7= 0.01 – 0.10 cos(x) + 0.01 cos (2x) + 0.04 cos(3x) - 0.01 cos(4x)

Fig 5.3(a) visualizes affine non-linear consequences which are generated by the proposed architecture. Local models can predict original outputs relatively well in the given neighborhood; in other words, this leads to forming a locally well interpretable model. Indeed, local models express good interaction with the global model, and each non-linear function depicts well the characteristics of the input-target relationship in the corresponding subset. Fig. 5.3(b) illustrates the overall output of the proposed model on the given data. Based on the figure, the model output is almost the same as the original output. The model's overall performance index (RMSE) is 0.0032 and 0.0034 for training and test data, respectively.



(a)



(b)

Figure 5. 3. Model's output on one-dimensional synthetic dataset #1: (a) local models and their corresponding prototypes, (b) overall model's output

Figure 5.4 illustrates the impact of changing the value of *c* (number of clusters) on the model's performance for both training and test data. This figure reveals an intuitively appealing tendency: the model's performance improves when the number of fuzzy rules increases. Furthermore, a plateau appears for the model performance after the number of clusters becomes seven. That is why we already set the number of clusters to *c*=7 in our experiments.

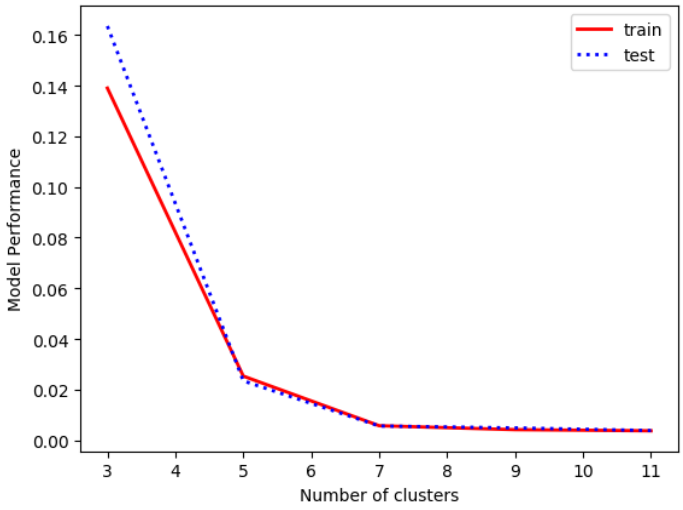


Figure 5. 4. Impact of number of clusters on the model's performance in the synthetic dataset #1

Figure 5.5 depicts the impact of the number of rules (clusters) on the TSK model's performance. This figure stresses that to achieve the desired accuracy, we need to increase the number of rules compared to our proposed nonlinear model. For instance, to obtain the same accuracy of 0.0032 as achieved by the non-linear consequences, the TSK model needs to have about seventeen rules rather than seven rules.

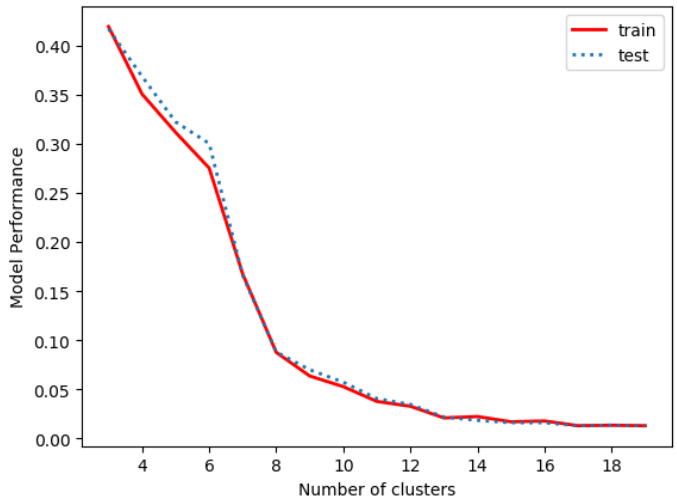


Figure 5. 5. Impact of number of clusters on the TSK's performance in the synthetic dataset #1

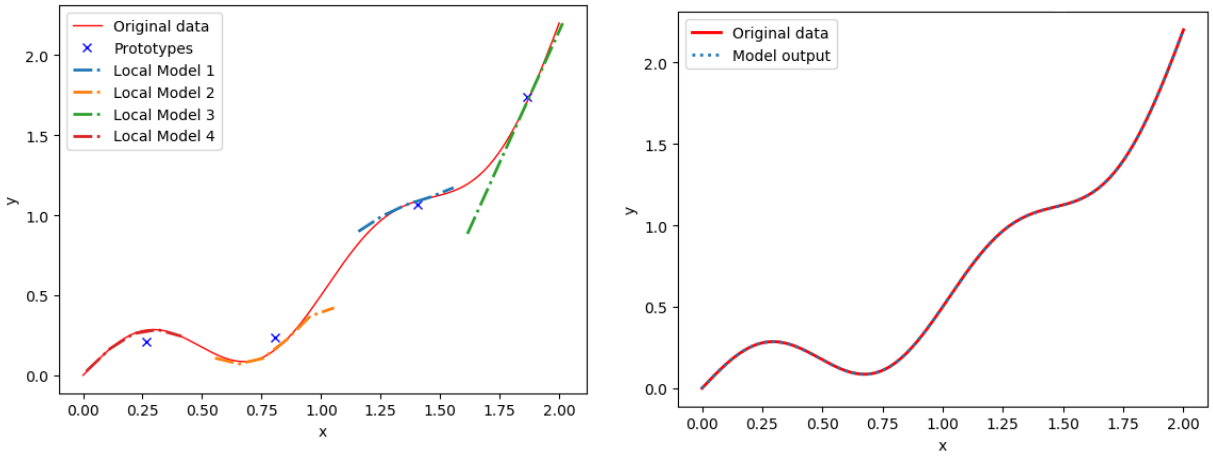
Figure 6.6 demonstrates the local models generated by the proposed architecture and the overall model's performance for synthetic dataset #2. The model's overall performance index is 0.0057 and 0.0058 for the training and test data, respectively. These values indicate that the model can predict the targets very well. The rules' consequences are a combination of non-linear (sine) terms for the first third rules, but linear function for the last rule. The consequences are still locally well interpretable and illustrate the input-target relationship in the corresponding subset. They come as follows:

f_1= 0.01+ 8.13 sin(x) - 0.53 sin (2x) - 8.03 sin (3x) + 4.58 sin(4x)

f_2= -0.14 - 2.62 sin(x) + 0.58 sin (2x) + 2.38 sin (3x) + 0.71 sin(4x)

f_3= 2.62 -2.39 sin(x) + 1.18 sin (2x) - 2.12 sin (3x) + 2.10 sin(4x)

f_4= -4.42 + 3.29 x



(a) (b)

Figure 5. 6. Model's output on synthetic dataset #2: (a) local models, and (b) overall model's output

Figures 5.7 and 5.8 illustrate the influence of changing the number of rules on the model's performance for training and testing data in synthetic dataset #2. There is a similar tendency, as we have already seen in Figure 5.4. The model's performance improves intuitively with the

increase in the number of rules. Comparison of Figure 5.7 and 5.8 also indicates that the TSK model needs more rules to achieve the same performance of the proposed model. For instance, to obtain the same accuracy of 0.0057 as achieved by the non-linear consequences, the TSK model needs to have nearly eighteen rules rather than four rules.

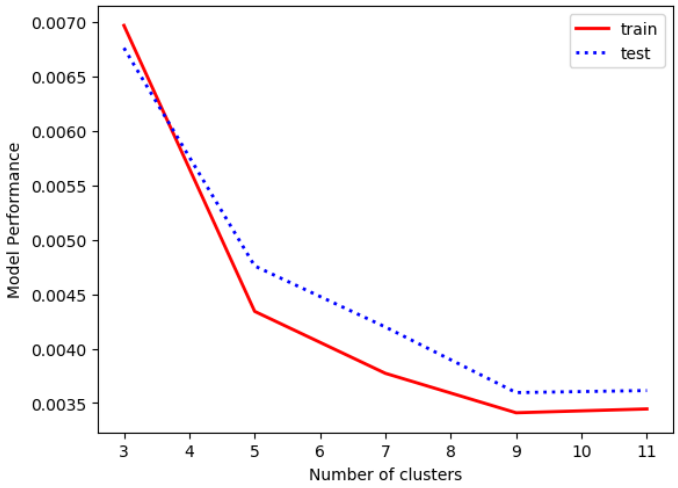


Figure 5. 7. Impact of number of clusters on the model's performance achieved in the synthetic dataset #2

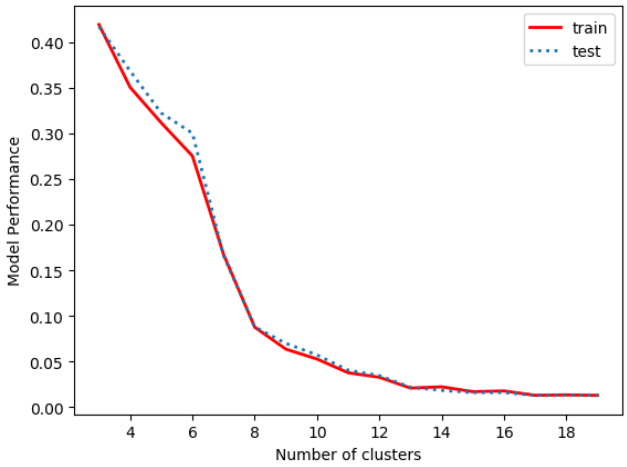
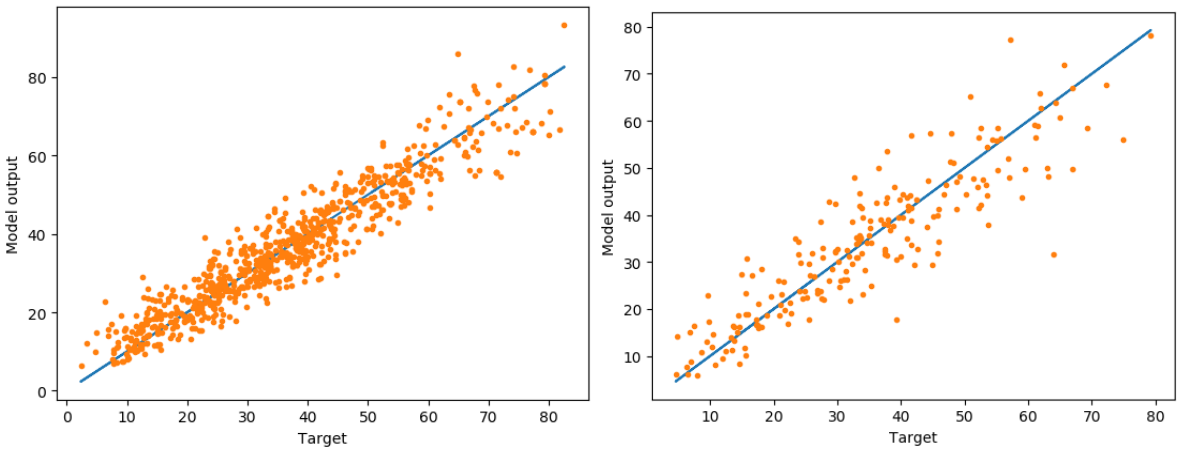


Figure 5. 8. Impact of number of clusters on the TSK model performance obtained in the synthetic dataset #2

5.4.2. Publicly available datasets

To further investigate the usefulness and quantify the performance of the introduced model, we perform experiments over publicly available real-world data sets. Some datasets, with different sizes and dimensions, from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml), the KEEL Datasets (https://sci2s.ugr.es/keel/datasets.php), the StatLib dataset, (http://lib.stat.cmu.edu/datasets), and the Bilkent University Function Approximation Repository (http://funapp.cs.bilkent.edu.tr/DataSets) are employed here. Similar to synthetic data sets, we use a 5-fold cross validation method to evaluate the models.

The concrete compressive strength is the first dataset from the UCI that is employed in our experiments. This dataset is a highly non-linear function of age and ingredients, and comprises 1030 instances with eight input variables and one real output. In this experiment, the proposed model's performance indices are measured for both train and test data and displayed in Figure 5. 9. The parameters used in this experiments after tuning are shown in Table 5.3.



a) Train data (b) Test data

Figure 5. 9. Performance of the proposed model on the concrete compressive strength dataset

Table 5.3. Parameters of the model

Parameter	Range	Selected value
Swarm size	[30-50]	50
Number of iterations	[20-50]	40
inertia weight damping ratio	[0.6-0.95]	0.8
c_1	{0.3,0.4,0.5,0.6, 0.7}	0.5
c_2	{0.3,0.4,0.5,0.6, 0.7}	0.5
m:Fuzzification coefficient	{1.7, 1.9, 2, 2.2, 2.5}	2

The consequences of rules which are generated by our proposed approach are some quadratic functions as follows:

$$f_1 = 0.01$$

$$+ 8.54\, x_1 - 11.3\, x_2 + 2.87 x_3 + 5.91 x_4 + 0.56\, x_5 - 21.8\, x_6 + 25.4\, x_7 + 1.56\, x_8$$

$$+ 0.02\, x_1^2 + 0.07\, x_1 x_2 - 0.005\, x_1 x_3 + 0.05\, x_1 x_4 + 0.06\, x_1 x_5 - 0.3\, x_1 x_6 - 0.05\, x_1 x_7$$

$$+ 0.3\, x_1 x_8 - 0.06\, x_2^2 + 0.03\, x_2 x_3 + 0.007\, x_2 x_4 + 0.02\, x_2 x_5 + 0.02\, x_2 x_6 - 0.00005\, x_2 x_7 - 0.004\, x_2 x_8$$

$$+ 0.002\, x_3^2 + 0.009\, x_3 x_4 + 0.007\, x_3 x_5 + 0.01\, x_3 x_6 + 0.01\, x_3 x_7 + 0.06\, x_3 x_8$$

$$+ 0.0001\, x_4^2 - 0.007\, x_4 x_5 + 0.002\, x_4 x_6 - 0.05\, x_4 x_7 + 0.02\, x_4 x_8$$

$$+ 0.01\, x_5^2 + 0.06\, x_5 x_6 - 0.07\, x_5 x_7 - 0.0004\, x_5 x_8$$

$$- 0.02\, x_6^2 - 0.006\, x_6 x_7 - 0.04\, x_6 x_8$$

$$+ 0.02\, x_7^2 + 0.06\, x_7 x_8$$

$$- 0.07\, x_8^2$$

f_2=-0.40

+ 0.006 x_1+ 0.01 x_2 + 0.008 x_3 + 0.28 x_4 + 0.0005 x_5 + 0.0009 x_6 - 0.0002 x_7 + 0.001 x_8

+ 0.002 x_1^2 -0.01 x_1 x_2 - 0.001 x_1 x_3 + 0.03 x_1 x_4 - 0.003 x_1 x_5 - 0.01 x_1 x_6 - 0.02 x_1 x_7 + 0.01 x_1 x_8

- 0.01 x_2^2 + 0.01 x_2 x_3 + 0.0010 x_2 x_4 - 0.03 x_2 x_5 +0.002 x_2 x_6 + 0.002 x_2 x_7 -0.01 x_2 x_8

+ 0.01 x_3^2 -0.001x_3 x_4 + 0.001x_3 x_5 + 0.03 x_3 x_6 + 0.001 x_3 x_7 -13.1 x_3 x_8

-16.8 x_4^2 + 14.06 x_4 x_5 +14.86 x_4 x_6- 0.02 x_4 x_7 + 13.01 x_4 x_8

- 15.43 x_5^2 - 5.57 x_5 x_6 + 0.002 x_5 x_7 +0.01 x_5 x_8

- 0.02 x_6 + 0.004 x_6 x_7 + 0.003 x_6 x_8

+ 0.001 x_7^2 + 0.01 x_7 x_8

+ 0.002 x_8^2

f_3= 0.01 - 0.01 x_1 + 0.01 x_2 + 0.01x_3 - 0.02 x_4 + 0.001 x_5 + 0.01 x_6 + 0.001 x_7 -0.001 x_8

+ 0.02 x_1^2 + 0.02 x_1 x_2 -0.01 x_1 x_3 - 0.03 x_1 x_4 +0.01 x_1 x_5 - 0.001 x_1 x_6 + 0.01 $x_1$$x_7$ + 0.003 x_1 x_8

- 0.01 x_2^2 - 0.01 x_2 x_3 - 0.01 x_2 x_4 + 0.04 x_2 x_5 - 0.01x_2 x_6 + 0.01 x_2 x_7 0.01 x_2 x_8

+ 0.002 x_3^2 + 0.009 x_3 x_4 + 0.007 x_3 x_5 + 0.28 x_3 x_6 + 0.01 x_3 x_7 + 0.06 x_3 x_8

+ 0.0001 x_4^2 - 0.007 x_4 x_5 + 0.002 x_4 x_6 - 0.05 x_4 x_7 + 0.02 x_4 x_8

+ 0.01 x_5^2 + 0.01 x_5 x_6 - 0.01 x_5 x_7 + 0.0001 x_5 x_8

+ 0.002 x_6^2 - 0.01 x_6 x_7 - 0.01 x_6 x_8

$+ 0.001\ x_7^2 + 0.01\ x_7\ x_8$

$- 0.01\ x_8^2$

$f_4 = 0.01 + 1.54\ x_1\ 2.1\ x_2 + 2.87 x_3 + 0.01 x_4 + 0.12\ x_5 - 0.15\ x_6 - 0.04\ x_7 + 1.56\ x_8$

We then build the standard TSK model with four linear consequences. These linear functions are as follows:

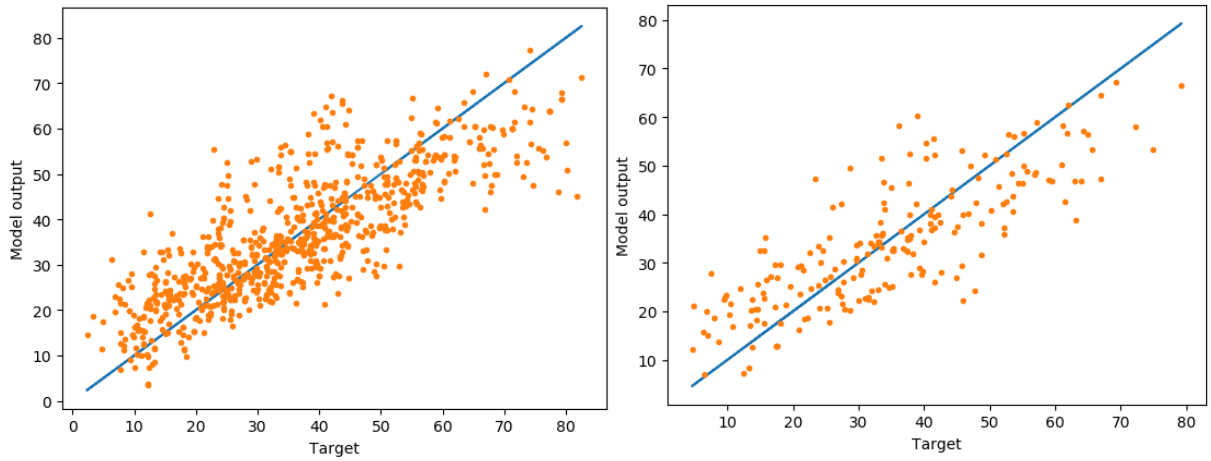
$f_1 = -260.28 - 0.008\ x_1 + 0.09\ x_2 + 0.04 x_3 + 0.49 x_4 + 1.69\ x_5 + 0.11\ x_6 + 0.08\ x_7 + 0.06\ x_8$

$f_2 = 166.82 - 0.06\ x_1 - 0.14\ x_2 - 0.17 x_3 - 0.34 x_4 - 0.26\ x_5 + 0.06\ x_6 - 0.15\ x_7 - 0.08\ x_8$

$f_3 = -644.66 + 0.12\ x_1 - 0.05 x_2 + 0.06 x_3 - 1.13\ x_4 - 1.83\ x_5 - 0.31\ x_6 - 0.13\ x_7 + 0.19\ x_8$

$f_4 = -687.30 + 0.42\ x_1 + 0.51 x_2 + 0.44 x_3 + 0.39\ x_4 + 1.23\ x_5 + 0.22\ x_6 + 0.31\ x_7 + 0.29\ x_8$

The results of the TSK model are also recorded in Figure 5.10. The results show that the proposed architecture improves the results compared to the standard TSK. The optimal structure's performance index obtained by the proposed model for the training data is 5.35 as opposed to 9.52 for the TSK model, and 7.27 versus 9.56 for the test data. This is intuitive because the proposed architecture benefits from the use of PSO, which optimizes the allocation of different non-linear and linear consequences rather than only using linear functions. If we want to improve the TSK model's performance index as the same as the proposed model, we need to increase the number of rules to nearly sixteen instead of four rules.



(a)Train data (b) Test data

Figure 5. 10. Performance of the TSK model on the concrete compressive strength dataset

Table 5.4 depicts the performance comparison (training and testing error) of the proposed model and standard TSK for various number of clusters (rules). This table indicates the improvement of our proposed model's performance with the optimized nonlinear and linear consequences compared to the standard TSK with only linear consequences. This table reveals that the number of rules to achieve the desired accuracy is lower in the proposed model compared to the standard TSK. In other words, the proposed modification can reduce the model's complexity while preserving the desired accuracy.

Other real-world are utilized to assess the effectiveness of arranging non-linear consequences in the rules. As datasets features come from different ranges, we do feature-wise normalization; for each feature in the input data, we subtract the mean of the feature and divide by the standard deviation so that the feature is centered on around zero and has a unit standard deviation. Along with the use of PSO as the optimization vehicle, we experiment with DE and GA that adopt the same scheme of coding candidate solutions that the PSO uses in Section 5.3. The numbers of

generations and the population's size are kept the same as those used in PSO (population size=20, and No. of iterations=20) to arrive at a sound comparative framework. We intend to compare the quality of results produced by the different methods and look at the methods' computational effectiveness, so we record a performance index (Q) and its running time (in seconds) for each model in Table 5.5. The number of clusters varies from 3 to 9. This table shows that the arrangement of non-linear consequences using the proposed evolutionary methods improves the results compared to the standard TSK. In other words, the structural optimization which is done by the proposed evolutionary methods improves the performance and reduces the model's complexity (number of rules) compared to the standard TSK model.

Although the performance index achieved by the proposed PSO-aided fuzzy model is not substantially different from the results of proposed DE and GA-aided models, from the computational point of view, PSO is still more efficient than DE and GA. Thus, it is concluded that the proposed model based on PSO achieves a balanced performance index and running time.

Table 5. 4. Performance comparison of the proposed model and standard TSK for the concrete compressive strength dataset (denoted by Mean ± Std. Dev. of 5 runs)

No. of rules	**Standard TSK**		**Proposed model**	
	Q_{train}	Q_{test}	Q_{train}	Q_{test}
c=2	$10.16 \pm 3\times10^{-3}$	$9.49 \pm 2\times10^{-2}$	$\mathbf{6.47 \pm 8\times10^{-2}}$	$\mathbf{7.57 \pm 2\times10^{-2}}$
c =3	$9.86 \pm 2\times10^{-5}$	$9.43 \pm 4\times10^{-2}$	$\mathbf{5.88 \pm 3\times10^{-2}}$	$\mathbf{8.03 \pm 1\times10^{-1}}$
c =4	$9.65 \pm 2\times10^{-5}$	$9.56 \pm 3\times10^{-1}$	$\mathbf{5.35 \pm 7\times10^{-1}}$	$\mathbf{7.26 \pm 2\times10^{-2}}$
c =5	$9.53 \pm 3\times10^{-3}$	$9.63 \pm 2\times10^{-1}$	$\mathbf{4.76 \pm 5\times10^{-2}}$	$\mathbf{7.98 \pm 5\times10^{-1}}$
c =6	$8.84 \pm 5\times10^{-2}$	$8.82 \pm 5\times10^{-2}$	$\mathbf{4.27 \pm 3\times10^{-3}}$	$\mathbf{7.10 \pm 2\times10^{-1}}$
c=7	$8.93 \pm 2\times10^{-5}$	$9.32 \pm 8\times10^{-2}$	$\mathbf{4.07 \pm 2\times10^{-2}}$	$\mathbf{7.11 \pm 5\times10^{-2}}$
c =8	$8.71 \pm 2\times10^{-5}$	$9.05 \pm 2\times10^{-4}$	$\mathbf{3.66 \pm 6\times10^{-3}}$	$\mathbf{8.00 \pm 2\times10^{-1}}$
c =9	$8.58 \pm 3\times10^{-4}$	$9.01 \pm 4\times10^{-4}$	$\mathbf{3.36 \pm 2\times10^{-2}}$	$\mathbf{8.01 \pm 8\times10^{-1}}$
c=10	$8.11 \pm 6\times10\text{-}^{2}$	$9.47 \pm 6\times10^{-2}$	$\mathbf{2.62 \pm 2\times10^{-2}}$	$\mathbf{8.83 \pm 8\times10^{-1}}$

Note: the entities in boldface represent the best performance obtained for two methods

Table 5. 5. Results of comparative analysis (denoted by Mean ± Std. Dev. of 5 runs)

Dataset	No. of rules	Standard TSK			Proposed PSO			Proposed DE			Proposed GA		
		Q		**Time (s)**	***Q***		**Time (s)**	***Q***		**Time (s)**	***Q***		**Time (s)**
		Train	**Test**		**Train**	**Test**		**Train**	**Test**		**Train**	**Test**	
Airfoil Self-Noise (1503*6)	*c*=3	$0.48\pm5\times10^{-5}$	$0.48\pm4\times10^{-5}$	0.20	$\mathbf{0.21\pm7\times10^{-6}}$	$\mathbf{0.12\pm8\times10^{-6}}$	3.79	$0.21\pm1\times10^{-5}$	$0.21\pm1\times10^{-5}$	31.49	$0.44\pm2\times10^{-2}$	$0.25\pm3\times10^{-2}$	19.48
	c=5	$0.44\pm2\times10^{-6}$	$0.44\pm2\times10^{-6}$	0.44	$\mathbf{0.18\pm7\times10^{-3}}$	$\mathbf{0.12\pm6\times10^{-3}}$	7.84	$0.18\pm6\times10^{-7}$	$0.20\pm1\times10^{-5}$	106.21	$0.44\pm2\times10^{-1}$	$0.21\pm2\times10^{-3}$	26.27
	c=7	$0.43\pm2\times10^{-6}$	$0.43\pm2\times10^{-6}$	0.84	$0.17\pm5\times10^{-2}$	$0.11\pm1\times10^{-2}$	10.76	$\mathbf{0.13\pm9\times10^{-7}}$	$\mathbf{0.15\pm1\times10^{-6}}$	287.64	$0.32\pm2\times10^{-2}$	$0.18\pm4\times10^{-3}$	21.74
	c=9	$0.37\pm2\times10^{-5}$	$0.38\pm1\times10^{-5}$	2.05	$0.13\pm5\times10^{-3}$	$0.06\pm3\times10^{-3}$	14.13	$\mathbf{0.11\pm1\times10^{-4}}$	$\mathbf{0.12\pm3\times10^{-2}}$	476.87	$0.34\pm1\times10^{-1}$	$0.14\pm1\times10^{-1}$	15.46
Yacht Hydrodynamics (308*7)	*c*=3	$0.40\pm5\times10^{-9}$	$0.37\pm1\times10^{-4}$	0.20	$\mathbf{0.08\pm3\times10^{-2}}$	$\mathbf{0.12\pm3\times10^{-2}}$	0.75	$0.09\pm3\times10^{-2}$	$0.14\pm4\times10^{-2}$	5.34	$0.15\pm6\times10^{-2}$	$0.14\pm5\times10^{-2}$	1.57
	c=5	0.33±0.0	$0.42\pm5\times10^{-3}$	0.18	$\mathbf{0.05\pm1\times10^{-3}}$	$\mathbf{0.10\pm1\times10^{-2}}$	1.09	$0.05\pm3\times10^{-4}$	$0.10\pm2\times10^{-3}$	13.13	$0.25\pm3\times10^{-1}$	$0.12\pm2\times10^{-2}$	1.92
	c=7	0.29±0.0	$0.56\pm3\times10^{-4}$	0.25	$\mathbf{0.04\pm2\times10^{-4}}$	$\mathbf{0.11\pm1\times10^{-2}}$	1.82	$0.04\pm2\times10^{-3}$	$0.11\pm2\times10^{-2}$	33.65	$0.20\pm3\times10^{-1}$	$0.10\pm1\times10^{-2}$	2.57
	c=9	0.30±0.0	$0.49\pm1\times10^{-3}$	0.25	$0.03\pm5\times10^{-3}$	$0.10\pm1\times10^{-2}$	3.87	$\mathbf{0.03\pm6\times10^{-4}}$	$\mathbf{0.05\pm4\times10^{-3}}$	104.88	$0.10\pm8\times10^{-2}$	$0.06\pm2\times10^{-3}$	2.76
Stock (950*10)	*c*=3	$0.17\pm1\times10^{-7}$	$0.18\pm1\times10^{-7}$	0.08	$\mathbf{0.08\pm1\times10^{-2}}$	$\mathbf{0.10\pm1\times10^{-7}}$	5.13	$0.08\pm7\times10^{-7}$	$0.10\pm5\times10^{-6}$	91.52	$0.16\pm7\times10^{-2}$	$0.10\pm4\times10^{-9}$	22.62
	c=5	$0.14\pm2\times10^{-3}$	$0.15\pm2\times10^{-3}$	0.14	$\mathbf{0.06\pm5\times10^{-3}}$	$\mathbf{0.07\pm1\times10^{-3}}$	9.43	$0.04\pm4\times10^{-3}$	$0.07\pm5\times10^{-3}$	261.50	$0.07\pm2\times10^{-1}$	$0.08\pm1\times10^{-2}$	25.46
	c=7	$0.13\pm2\times10^{-3}$	$0.14\pm5\times10^{-3}$	0.20	$\mathbf{0.03\pm3\times10^{-3}}$	$\mathbf{0.07\pm1\times10^{-3}}$	15.88	$0.03\pm7\times10^{-4}$	$0.08\pm3\times10^{-3}$	241.22	$0.05\pm1\times10^{-1}$	$0.08\pm5\times10^{-3}$	24.74
	c=9	$0.12\pm8\times10^{-7}$	$0.14\pm2\times10^{-6}$	0.19	$\mathbf{0.02\pm3\times10^{-3}}$	$\mathbf{0.09\pm2\times10^{-3}}$	21.53	$0.02\pm1\times10^{-3}$	0.10±±3×10-3	325.35	$0.07\pm1\times10^{-1}$	$0.11\pm1\times10^{-2}$	28.12
Baseball (337*17)	*c*=3	$0.45\pm3\times10^{-8}$	$0.54\pm2\times10^{-7}$	0.10	$0.49\pm5\times10^{-8}$	$0.50\pm2\times10^{-3}$	7.01	**0.43±5×10-4**	$\mathbf{0.52\pm4\times10^{-2}}$	6.93	$0.73\pm2\times10^{-1}$	$0.50\pm1\times10^{-3}$	2.17
	c=5	$0.38\pm6\times10^{-3}$	$0.53\pm2\times10^{-3}$	0.23	$\mathbf{0.24\pm2\times10^{-1}}$	$\mathbf{0.49\pm1\times10^{-2}}$	19.25	0.28±2×10-6	$0.50\pm1\times10^{-6}$	26.22	$0.25\pm2\times10^{-1}$	$0.49\pm1\times10^{-3}$	5.27
	c=7	$0.33\pm3\times10^{-3}$	$0.63\pm1\times10^{-2}$	0.25	$\mathbf{4\times10^{-3}\pm0.02}$	$\mathbf{0.55\pm2\times10^{-3}}$	28.49	$0.04\pm2\times10^{-1}$	$0.56\pm5\times10^{-1}$	176.05	$0.04\pm1\times10^{-1}$	$0.57\pm2\times10^{-2}$	138.41
	c=9	$0.31\pm4\times10^{-3}$	$0.56\pm2\times10^{-2}$	0.37	$\mathbf{3\times10^{-5}\pm0.01}$	$\mathbf{0.51\pm7\times10^{-3}}$	78.83	$3\times10^{-3}\pm0.02$	$1.03\pm3\times10^{-2}$	650.30	$0.10\pm2\times10^{-1}$	$0.89\pm7\times10^{-2}$	51.16
Body Fat (252*15)	*c*=3	$0.49\pm3\times10^{-7}$	$0.56\pm2\times10^{-6}$	0.01	$\mathbf{0.42\pm3\times10^{-6}}$	$\mathbf{0.52\pm1\times10^{-7}}$	3.21	$0.44\pm5\times10^{-3}$	$0.52\pm5\times10^{-2}$	7.81	$0.45\pm2\times10^{-1}$	$0.52\pm2\times10^{-2}$	1.12
	c=5	$0.42\pm2\times10^{-8}$	$0.68\pm2\times10^{-8}$	0.03	$0.40\pm7\times10^{-3}$	$0.54\pm2\times10^{-1}$	8.56	**0.39±4×10-1**	$\mathbf{0.62\pm5\times10^{-1}}$	130.76	0.39±3×10-1	$0.62\pm5\times10^{-1}$	12.54
	c=7	$0.38\pm3\times10^{-5}$	$0.86\pm4\times10^{-4}$	0.02	$\mathbf{0.05\pm2\times10^{-8}}$	$\mathbf{0.65\pm8\times10^{-2}}$	15.02	$0.10\pm5\times10^{-1}$	$0.69\pm7\times10^{-2}$	140.44	$0.10\pm2\times10^{-1}$	$0.69\pm4\times10^{-2}$	3.01
	c=9	$0.33\pm8\times10^{-4}$	$0.98\pm1\times10^{-2}$	0.04	$\mathbf{4\times10^{-5}\pm0.01}$	$\mathbf{0.90\pm3\times10^{-2}}$	21.19	$0.01\pm9\times10^{-2}$	$0.93\pm2\times10^{-2}$	330.47	$0.03\pm9\times10^{-2}$	$0.95\pm2\times10^{-2}$	48.41

Boston Housing (506*14)	c=3	$0.34\pm7\times10^{-7}$	$0.48\pm2\times10^{-6}$	0.01	$\mathbf{0.12\pm5\times10^{-3}}$	$\mathbf{0.44\pm6\times10^{-3}}$	2.89	$0.12\pm3\times10^{-3}$	$0.44\pm6\times10^{-3}$	9.07	$0.18\pm2\times10^{-1}$	$0.42\pm1\times10^{-2}$	0.78
	c=5	$0.30\pm3\times10^{-5}$	$0.48\pm1\times10^{-4}$	0.02	$\mathbf{0.12\pm8\times10^{-2}}$	$\mathbf{0.53\pm6\times10^{-3}}$	7.28	$0.15\pm5\times10^{-2}$	$0.53\pm2\times10^{-3}$	154.58	$0.15\pm2\times10^{-1}$	$0.13\pm2\times10^{-6}$	1.12
	c=7	$0.28\pm1\times10^{-5}$	$0.51\pm4\times10^{-6}$	0.05	$\mathbf{0.12\pm8\times10^{-2}}$	$\mathbf{0.68\pm1\times10^{-2}}$	11.12	$0.12\pm3\times10^{-2}$	$0.68\pm3\times10^{-2}$	148.34	$0.12\pm2\times10^{-1}$	$0.62\pm5\times10^{-3}$	2.24
	c=9	$0.25\pm3\times10^{-3}$	$0.52\pm1\times10^{-2}$	0.06	$\mathbf{2\times10^{-8}\pm0.08}$	$\mathbf{0.51\pm6\times10^{-2}}$	16.94	$0.15\pm2\times10^{-2}$	$0.50\pm4\times10^{-2}$	51.54	$0.10\pm2\times10^{-1}$	$0.49\pm2\times10^{-2}$	15.80
Weather Izmir (1461*10)	c=3	$0.08\pm5\times10^{-9}$	$0.07\pm1\times10^{-7}$	0.02	$0.07\pm4\times10^{-9}$	$0.08\pm1e^{-7}$	8.09	$\mathbf{0.04\pm1\times10^{-8}}$	$\mathbf{0.06\pm2\times10^{-8}}$	68.88	$0.15\pm1\times10^{-1}$	$0.05\pm3\times10^{-7}$	17.21
	c=5	$0.08\pm3\times10^{-8}$	$0.06\pm3\times10^{-8}$	0.07	$0.07\pm1\times10^{-3}$	$0.07\pm4\times10^{-3}$	30.68	$\mathbf{0.03\pm\times10^{-6}}$	$\mathbf{0.05\pm2\times10^{-5}}$	391.91	$0.23\pm2\times10^{-1}$	$0.05\pm6\times10^{-4}$	32.10
	c=7	$0.07\pm3\times10^{-7}$	$0.06\pm3\times10^{-6}$	0.19	$0.07\pm2\times10^{-3}$	$0.09\pm2\times10^{-3}$	37.70	$\mathbf{0.03\pm2\times10^{-3}}$	$\mathbf{0.07\pm3\times10^{-3}}$	743.28	$0.18\pm2\times10^{-1}$	$0.06\pm1\times10^{-3}$	38.01
	c=9	$0.07\pm4\times10^{-5}$	$0.07\pm2\times10^{-5}$	0.14	$0.08\pm8\times10^{-4}$	$0.07\pm8\times10^{-3}$	36.69	$0.03\pm4\times10^{-3}$	$0.07\pm5\times10^{-4}$	1410.84	$\mathbf{0.03\pm8\times10^{-2}}$	$\mathbf{0.06\pm3\times10^{-5}}$	44.89
Auto MPG (398*7)	c=3	$0.57\pm1\times10^{-4}$	$0.53\pm8\times10^{-5}$	0.01	$0.45\pm5\times10^{-2}$	0.52 ± 0.04	3.47	$\mathbf{0.33\pm7\times10^{-6}}$	$\mathbf{0.51\pm9\times10^{-6}}$	8.5	$0.51\pm2\times10^{-1}$	$0.51\pm7\times10^{-5}$	7.16
	c=5	$0.39\pm2\times10^{-2}$	$0.39\pm1\times10^{-2}$	0.01	$0.27\pm4\times10^{-2}$	0.37 ± 0.05	4.99	$\mathbf{0.14\pm1\times10^{-3}}$	$\mathbf{0.29\pm1\times10^{-3}}$	27.82	$0.34\pm2\times10^{-1}$	$0.27\pm1\times10^{-2}$	7.99
	c=7	$0.35\pm5\times10^{-3}$	$0.35\pm2\times10^{-3}$	0.03	$0.27\pm6\times10^{-2}$	0.36 ± 0.07	6.66	$\mathbf{0.27\pm2\times10^{-2}}$	$\mathbf{0.25\pm2\times10^{-2}}$	63.39	$0.35\pm2\times10^{-1}$	$0.24\pm6\times10^{-2}$	9.17
	c=9	$0.28\pm1\times10^{-2}$	$0.30\pm7\times10^{-3}$	0.03	$\mathbf{0.24\pm4\times10^{-2}}$	$\mathbf{0.34\pm0.08}$	8.86	$0.25\pm2\times10^{-2}$	$0.48\pm2\times10^{-2}$	82.66	$0.33\pm2\times10^{-1}$	$0.42\pm1\times10^{-2}$	6.02
Forest Fires (517*13)	c=3	$1.04\pm5\times10^{-6}$	$0.46\pm3\times10^{-4}$	0.01	$\mathbf{1.02\pm4\times10^{-2}}$	$\mathbf{0.31\pm1\times10^{-2}}$	7.20	$1.02\pm3\times10^{-2}$	$0.31\pm4\times10^{-4}$	6.93	$1.02\pm5\times10^{-6}$	$0.30\pm7\times10^{-4}$	2.47
	c=5	$1.04\pm8\times10^{-6}$	$0.59\pm1\times10^{-2}$	0.01	$\mathbf{1.10\pm6\times10^{-7}}$	$\mathbf{0.31\pm7\times10^{-6}}$	12.50	$1.02\pm3\times10^{-3}$	$0.31\pm5\times10^{-4}$	26.22	$1.02\pm8\times10^{-6}$	$0.31\pm9\times10^{-4}$	2.66
	c=7	$1.04\pm7\times10^{-6}$	$0.59\pm1\times10^{-2}$	0.01	$\mathbf{1.02\pm3\times10^{-3}}$	$\mathbf{0.31\pm1\times10^{-2}}$	20.51	$1.02\pm5\times10^{-2}$	$0.31\pm3\times10^{-4}$	176.05	$1.02\pm7\times10^{-6}$	$0.31\pm7\times10^{-4}$	3.59
	c=9	$1.04\pm9\times10^{-6}$	$0.59\pm1\times10^{-2}$	0.02	$1.02\pm4\times10^{-3}$	$0.36\pm1\times10^{-2}$	21.70	$\mathbf{1.02\pm5\times10^{2}}$	$\mathbf{0.31\pm3\times10^{-4}}$	650.30	$1.02\pm9\times10^{-6}$	$0.31\pm6\times10^{-4}$	2.55

Note: the entities in boldface represent the best performance obtained for four methods

5.5. Conclusions

In this chapter, an identification framework has been proposed for fuzzy rule-based models. This framework stresses the need to use non-linear functions (rather than linear) standing in the fuzzy rules' consequences. The proposed architecture benefits from PSO and LSE to extract the consequences, whereas fuzzy sets standing in the antecedents of rules are formed by the FCM clustering algorithm. In this integrated architecture, the TSK model is developed based on the allocation of diverse non-linear functions such as sine, cosine, square root, and exponential functions as well as linear functions to form the consequences of rules. We applied our proposed fuzzy model to some synthetic and real-world datasets with different sizes and dimensions and then compared it with the standard TSK model with linear consequences. The experimental results indicate some improvements in the performance of the introduced fuzzy model compared to the standard TSK's performance. The results also reveal that the number of rules to achieve the desired accuracy is lower in the proposed model than the number of rules in standard TSK. In other words, the proposed modification can reduce the model's complexity preserving the desired accuracy.

Chapter 6 Conclusions and Future Studies

This dissertation is concerned with the development and analysis of fuzzy models using population-based algorithms. Three different architectures have been proposed, and these architectures focused on the need for structural and parametric optimizations of fuzzy models, particularly rule-based models, to make a less complex but accurate model. Overall, the approaches introduced in this dissertation exhibited some impressive highlights. This chapter briefly summarizes the significant contributions and points out some interesting research topics for future studies.

6.1. Major Contributions

In the first work of this thesis, an identification framework for fuzzy rule-based models has been developed. This framework stresses the need for and benefits of structural refinement of fuzzy rules to make the model less complex while still retaining its accuracy. Two different ways of structuralizing the antecedents and consequences of the rules, based on the newly introduced modeling resources, are proposed: (i) the arrangement of input spaces in the antecedents of the rules and (ii) the arrangement of the orders of polynomials in the consequences of the rules. The modeling resources are concerned with (i) the total order of polynomials encountered across all the rules and (ii) the fraction of overall number of input variables of the original space. A hybrid methodology is proposed in which the PSO guided by RMSE accuracy criterion is employed to find the efficient arrangements of input spaces and orders of polynomials. In this method, fuzzy sets standing in the antecedents of rules are created by the FCM, while the coefficients of the

polynomials are estimated by the standard LSE method. Different TSK models formed in this study are based on the optimization of antecedents or consequences of fuzzy rules. The experimental studies, involving synthetic datasets and some well-known datasets coming from the UCI, Bilkent, and KEEL repositories, demonstrate the dominance of the model with optimal allocation of orders of polynomials in consequences over the reduction of input space in antecedents of fuzzy rules in most of the datasets. Based on the experiments, simultaneous optimization of both antecedents and consequences of fuzzy rules is also a promising avenue of creating fuzzy models with respect to both complexity and accuracy criteria. The performance of the model is also influenced by some crucial parameters of the model, such as the fraction of original input space employed in the model.

In the second work, a GP-based architecture has been proposed for the automatic extraction of the fuzzy models from input-output data. In the proposed method, the structure of a fuzzy model is represented by a multi-tree individual and GP is utilized to find the optimal fuzzy model. The idea of admitting augmented fuzzy operators (e.g. Xor, Nor, Nand, Equivalence, and Implication) in addition to basic operators (viz., *t*-norm, *t*-conorm and negation) is also introduced, and then the effects of using different *t*-norms and *t*-conorms on model performance is discussed. This study also addresses the distance concentration issue in the design of high-dimensional fuzzy models, where the increase of dimensionality may converge all the pairwise distances (dissimilarities) to the same value, leading to failure of the fuzzy model. In this study, FCM is applied independently to each variable, rather than to all variables (dimensions) at once, to eliminate the problem of distance concentration. Subsequently, the structural optimization of the model is accomplished using multi-tree genetic programming, and parametric optimization is performed via gradient-based learning. We applied our GP-based fuzzy model to synthetic datasets; specifically, some

regression datasets from the Penn Machine Learning Benchmarks and the ICOS PSP Benchmarks Repository, and we compared our methodology with other approaches. In the experiments, our GP-based model (with the augmented logic operators) attained higher performance compared to the GP model with the basic logic operators, and the standard TSK fuzzy model available in the literature.

In the third work, an identification framework has been proposed for fuzzy rule-based models. This framework stresses the need to use non-linear functions (rather than linear) standing in the fuzzy rules' consequences. The proposed architecture benefits from PSO and LSE to extract the consequences, whereas fuzzy sets standing in the antecedents of rules are formed by the FCM clustering algorithm. In this integrated architecture, the TSK model is developed based on the allocation of diverse non-linear functions such as sine, cosine, square root, and exponential functions as well as linear functions to form the consequences of rules. We applied our proposed fuzzy model to some synthetic and real-world datasets with different sizes and dimensionality, and then the model is compared with the standard TSK model which uses linear consequences. The experimental results indicate some improvements in the prospered fuzzy model's performance compared to the standard TSK fuzzy model. The results also reveal that the number of rules to achieve the desired accuracy is lower in the proposed model compared to the standard TSK. In other words, the proposed modification can reduces the model’s complexity retaining the desired accuracy.

6.2. Future Studies

Although many interesting and essential topics have been investigated so far, we point out that there are still many ideas worth further investigating. We list several directions which are of

interest to be explored in our future studies.

(1) ***Structure discovery of fuzzy models using logic minimization***

The idea in this work is using binary logic minimization in the structure discovery of fuzzy models. The essence is to consider numeric data, transform them to logic-based (Boolean) counterparts, develop logic–oriented mapping (Boolean functions), and finally convert logic results to numeric equivalents. The crux is to cast the problem in its underlying logic setting and exploit well-known algebraic methods in digital systems to minimize Boolean functions. By understanding the logical nature of real-world data from the beginning of the design, they can reveal a structure that can produce accurate and highly interpretable models.

(2) ***Structural Optimization of evolvable fuzzy models***

In the real world, we often encounter systems that change over time and/or space. The systems (models) have to evolve as the real world is often non-stationary. An essential category of non-stationary systems concerns situations when data behind the model are coming in temporal segments (week, month, year, etc.) so that the resulting model evolves from one time segment to another. In processing temporal segments of multi-variable data such as data coming from sensor networks, stock market, currency exchange rate, etc., readings are analyzed over some time slices, and for each time slice, a fuzzy rule-based model is constructed. As data could fluctuate by moving up and down in consecutive time segments, the models built in successive steps can change their granularity (level of detail). In other words, as fuzzy models are mainly created based on information granulation (clusters), the number of information granules can either increase or decrease to reflect the varying complexity of streams of data. This change of clusters is carried out by merging clusters (local models) with solid performance or splitting the clusters with weak performance.

Other types of structural optimization of fuzzy models for each data segment can be completed to increase the interpretability of models. The number of fuzzy sets, type of membership functions, and type of aggregation operators in the fuzzy models could be optimized using global optimization methods like evolutionary algorithms. The importance of such models becomes more evident in complex system modeling, especially in case of problems of high-dimensionality. This project's fundamental objective is to develop a new design methodology of evolvable fuzzy rule-based models based on the principles of information granulation and evolutionary algorithms for predicting aims, stressing on the increase of interpretability and approximation performance and striking a sound compromise between these two requirements.

(3) ***Dealing with high-dimensional data with the development of accurate-interpretable fuzzy model with integration of soft subspace clustering and evolutionary methods***

This work's motivation is to form a fuzzy model that strikes a sound balance between accuracy and interpretability. Soft subspace clustering (SSC) is used to extract the optimal partition of input space to obtain important feature subsets for different clusters. This kind of clustering allows the fuzzy model to possess a variable number of features in each rule. With the SSC results, which form the antecedents of fuzzy rules, the orders and parameters of polynomials in consequence of fuzzy rules are determined by an evolutionary algorithm and minimizing LSE optimization criterion, respectively. On the one hand, feature selection over the antecedents of rules and efficient arrangement of polynomials' orders in the consequences leads to an interpretable model.

On the other hand, this process is driven by the RMSE accuracy criterion treated as a fitness function of the evolutionary algorithm. Thus, both the accuracy and complexity of the model are enhanced. Due to the reduction of input spaces by SCC, this architecture can also be utilized to form high-dimensional data-driven TSK models.

(4) ***Design of Explainable Deep Neural Networks Using Fuzzy Computing***

Deep neural networks (DNN) have become the most effective approach to a wide variety of industry domains, including image recognition tasks, natural language processing, speech recognition, and many others; however, those models remain incomprehensible black boxes, composed of millions of connections in several layers. This makes the model uninterpretable for humans, and mostly the users are not willing to trust such a model for critical decisions, such as health care, finance, so the need for an Explainable AI becomes essential.

On the other hand, fuzzy rule-based models are by design much more interpretable. By introducing an additional machine learning process, fuzzy inference, an explainable rule-based structure can be realized in Deep Learning (DL) systems alleviating the lack of interpretability issue. The fuzzy inference step allows a user to generate rule-based structures. By creating these rules, it is possible for the user (analyst) to understand better the features developed by the DL system. The importance of such models becomes more evident in complex system modeling, especially in case of problems of high-dimensionality.

Bibliography

[1] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, 1985.

[2] A. Fernández, V. López, M. J. D. Jesus, and F. Herrera, "Revisiting Evolutionary Fuzzy Systems: Taxonomy, applications, new trends and challenges," *Knowledge-Based Systems*, vol. 80, pp. 109–121, 2015.

[3] V. Ojha, A. Abraham, and V. Snášel, "Heuristic design of fuzzy inference systems: A review of three decades of research," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 845–864, 2019.

[4] N. L. Tsakiridis, J. B. Theocharis, and G. C. Zalidis, "DECO 3 RUM: A Differential Evolution learning approach for generating compact Mamdani fuzzy rule-based models," *Expert Systems with Applications*, vol. 83, pp. 257–272, 2017.

[5] A. S. Koshiyama, R. Tanscheit, and M. M. B. R. Vellasco, "Automatic synthesis of fuzzy systems: An evolutionary overview with a genetic programming perspective," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 2, 2018.

[6] W. Pedrycz and F. Gomide, Fuzzy systems engineering: toward human-centric computing. Hoboken, NJ: John Wiley, 2007.

[7] F. Klawonn, "What Can Fuzzy Cluster Analysis Contribute to Clustering of High-Dimensional Data?," *Fuzzy Logic and Applications Lecture Notes in Computer Science*, pp. 1–14, 2013.

[8] S. Kumari and B. Jayaram, "Measuring Concentration of Distances—An Effective and Efficient Empirical Index," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 373–386, 2017.

[9] J. C. Bezdek, R. Ehrlich, and W. Full, “FCM: The fuzzy c-means clustering algorithm,” *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[10] E. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[11] A. P. Engelbrecht, Computational intelligence: An introduction. Chichester, West Sussex, England: John Wiley, 2007.

[12] J. Koza, “Genetic programming as a means for programming computers by natural selection,” *Statistics and Computing*, vol. 4, no. 2, 1994.

[13] M. Setnes and H. Roubos, “GA-fuzzy modeling and classification: complexity and performance,” *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 509–522, 2000.

[14] M. Setnes, R. Babuska, U. Kaymak, and H. V. N. Lemke, “Similarity measures in fuzzy rule base simplification,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 376–386, 1998.

[15] C.-C. Chen and C.-C. Wong, “A GA-based method for constructing fuzzy systems directly from numerical data,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 30, no. 6, pp. 904–911, 2000.

[16] Y.-W. Teng and W.-J. Wang, “Constructing a User-Friendly GA-Based Fuzzy System Directly From Numerical Data,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2060–2070, 2004.

[17] W. Pedrycz and M. Song, “A genetic reduction of feature space in the design of fuzzy models,” *Applied Soft Computing*, vol. 12, no. 9, pp. 2801–2816, 2012.

[18] J.-N. Choi, S.-K. Oh, and W. Pedrycz, “Identification of fuzzy models using a successive tuning method with a variant identification ratio,” *Fuzzy Sets and Systems*, vol. 159, no. 21, pp. 2873–2889, 2008.

[19] J.-N. Choi, S.-K. Oh, and W. Pedrycz, “Structural and parametric design of fuzzy inference systems using hierarchical fair competition-based parallel genetic algorithms and information granulation,” *International Journal of Approximate Reasoning*, vol. 49, no. 3, pp. 631–648, 2008.

[20] C. Li and T. Wu, “Adaptive fuzzy approach to function approximation with PSO and RLSE,” *Expert Systems with Applications*, vol. 38, no. 10, pp. 13266–13273, 2011.

[21] N. L. Tsakiridis, J. B. Theocharis, and G. C. Zalidis, “DECO 3 RUM: A Differential Evolution learning approach for generating compact Mamdani fuzzy rule-based models,” *Expert Systems with Applications*, vol. 83, pp. 257–272, 2017.

[22] S.-H. Tsai and Y.-W. Chen, “A novel identification method for Takagi–Sugeno fuzzy model,” *Fuzzy Sets and Systems*, vol. 338, pp. 117–135, 2018.

[23] N. L. Tsakiridis, J. B. Theocharis, P. Panagos, and G. C. Zalidis, “An evolutionary fuzzy rule-based system applied to the prediction of soil organic carbon from soil spectral libraries,” *Applied Soft Computing*, vol. 81, p. 105504, 2019.

[24] H. Ishibuchi and Y. Nojima,” Discussions on interpretability of fuzzy systems using simple examples”, In: Proceedings of IFSA-EUSFLAT World Congress, pp. 1649-1654, 2009.

[25] M. Antonelli, P. Ducange, B. Lazzerini, and F. Marcelloni, “Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity,” *Soft Computing*, vol. 15, no. 12, pp. 2335–2354, 2010.

[26] J. Casillas, P. Martínez, and A. D. Benítez, "Learning consistent, complete and compact sets of fuzzy rules in conjunctive normal form for regression problems," *Soft Computing*, vol. 13, no. 5, pp. 451–465, 2008.

[27] H. Ishibuchi and Y. Nojima, "Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning," *International Journal of Approximate Reasoning*, vol. 44, no. 1, pp. 4–31, 2007.

[28] P. Pulkkinen, J. Hytönen, and H. Koivisto, "Developing a bioaerosol detector using hybrid genetic fuzzy systems," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1330–1346, 2008.

[29] H. Ishibuchi, T. Murata, and I. Türkşen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135–150, 1997.

[30] L. Booker, Intelligent behavior as an adaption to the task environment (Doctoral. dissertation). University of Michigan, Ann Arbor, U.S., 1982.

[31] S. Smith, A learning system based on genetic adaptive algorithms (Doctoral. dissertation). University of Pittsburgh, Pittsburgh, U.S, 1980.

[32] A. Lahsasna and W. C. Seng, "An improved genetic-fuzzy system for classification and data analysis," *Expert Systems with Applications*, vol. 83, pp. 49–62, 2017.

[33] A. A. Marquez, F. A. Marquez, and A. Peregrin, "A multi-objective evolutionary algorithm with an interpretability improvement mechanism for linguistic fuzzy systems with adaptive defuzzification," *International Conference on Fuzzy Systems*, 2010.

[34] M. Fazzolari, B. Giglio, R. Alcalá, F. Marcelloni, and F. Herrera, "A study on the application of instance selection techniques in genetic fuzzy rule-based classification systems: Accuracy-complexity trade-off," *Knowledge-Based Systems*, vol. 54, pp. 32–41, 2013.

[35] M. J. Gacto, R. Alcalá, and F. Herrera, "Integration of an Index to Preserve the Semantic Interpretability in the Multiobjective Evolutionary Rule Selection and Tuning of Linguistic Fuzzy Systems," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 3, pp. 515–531, 2010.

[36] H. H. Y. Saad, N. A. M. Isa, M. M. Ahmed, and A. H. Y. Sad, "A robust structure identification method for evolving fuzzy system," *Expert Systems with Applications*, vol. 93, pp. 267–282, 2018.

[37] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, and F. Herrera, "A Review of the Application of Multiobjective Evolutionary Fuzzy Systems: Current Status and Further Directions," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 45–65, 2013.

[38] O. Cordón, "A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems," *International Journal of Approximate Reasoning*, vol. 52, no. 6, pp. 894–913, 2011.

[39] F. Berlanga, A. Rivera, M. D. Jesus, and F. Herrera, "GP-COACH: Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems," *Information Sciences*, vol. 180, no. 8, pp. 1183–1200, 2010.

[40] A. S. Koshiyama, M. M. Vellasco, and R. Tanscheit, "GPFIS-CLASS: A Genetic Fuzzy System based on Genetic Programming for classification problems," *Applied Soft Computing*, vol. 37, pp. 561–571, 2015.

[41] W. Pedrycz and M. Reformat, “Evolutionary fuzzy modeling,” *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 5, pp. 652–665, 2003.

[42] Z. Liu, C. L. P. Chen, Y. Zhang, and H.-X. Li, “Type-2 hierarchical fuzzy system for high-dimensional data-based modeling with uncertainties,” *Soft Computing*, vol. 16, no. 11, pp. 1945–1957, 2012.

[43] E. K. Aydogan, I. Karaoglan, and P. M. Pardalos, “hGA: Hybrid genetic algorithm in fuzzy rule-based classification systems for high-dimensional problems,” *Applied Soft Computing*, vol. 12, no. 2, pp. 800–806, 2012.

[44] H. Ishibuchi and T. Yamamoto, “Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining,” *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 59–88, 2004.

[45] O. Cordon, F. Herrera, and P. Villar, “Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base,” *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 667–674, 2001.

[46] R. Ji, Y. Yang, and W. Zhang, “TS-fuzzy modeling based on ε-insensitive smooth support vector regression,” *Journal of Intelligent & Fuzzy Systems*, vol. 24, no. 4, pp. 805–817, 2013.

[47] J. Zhang, Z. Deng, K.-S. Choi, and S. Wang, “Data-Driven Elastic Fuzzy Logic System Modeling: Constructing a Concise System With Human-Like Inference Mechanism,” *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, pp. 2160–2173, 2018.

[48] L. Jing, M. K. Ng, and J. Z. Huang, "An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1026–1041, 2007.

[49] Z. Deng, K.-S. Choi, F.-L. Chung, and S. Wang, "Enhanced soft subspace clustering integrating within-cluster and between-cluster information," *Pattern Recognition*, vol. 43, no. 3, pp. 767–781, 2010.

[50] L. Liu, D. Tao, and Q. C. Amp, "Review on recent method of solving lasso problem," J. Data Acquis. Process.,vol. 30, no. 1, pp. 35–46, 2015.

[51] P. Xu, Z. Deng, C. Cui, T. Zhang, K.-S. Choi, S. Gu, J. Wang, and S. Wang, "Concise Fuzzy System Modeling Integrating Soft Subspace Clustering and Sparse Learning," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 11, pp. 2176–2189, 2019.

[52] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, A field guide to genetic programming. Lieu de publication non identifié: Lulu Press, 2008.

[53] G. Zhong and C.-M. Pun, "Subspace clustering by simultaneously feature selection and similarity learning," *Knowledge-Based Systems*, vol. 193, p. 105512, 2020.

[54] K. Tanaka, H. Yoshida, H. Ohtake, and H. O. Wang, "A sum-of-squares approach to modeling and control of nonlinear dynamical systems with polynomial fuzzy systems," *IEEE Transactions on Fuzzy Systems*., vol. 17, no. 4, pp. 911–922, 2009.

[55] A. Sala, "On the conservativeness of fuzzy and fuzzy-polynomial control of nonlinear systems," Annual Reviews in Control, vol. 33, no. 1, pp. 48–58, 2009.

[56] J. Dong, Y. Wang, and G.-H. Yang, "Control Synthesis of Continuous-Time T-S Fuzzy

Systems With Local Nonlinear Models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 5, pp. 1245–1258, 2009.

[57] H. Moodi, M. Farrokhi, T. M. Guerra, and J. Lauber, "On Stabilization Conditions for T–S Systems with Nonlinear Consequent Parts," *International Journal of Fuzzy Systems*, vol. 21, no. 1, pp. 84–94, 2018.

[58] R. F. Araújo, P. H. Coutinho, A.-T. Nguyen, and R. M. Palhares, "Delayed nonquadratic L2-stabilization of continuous-time nonlinear Takagi–Sugeno fuzzy models," *Information Sciences*, vol. 563, pp. 59–69, 2021.

[59] R. F. Araujo, L. A. B. Torres, and R. M. Palhares, "Distributed Control of Networked Nonlinear Systems via Interconnected Takagi-Sugeno Fuzzy Systems with Nonlinear Consequent," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–10, 2019.

[60] L. Teng, Y. Wang, W. Cai, and H. Li, "Robust Fuzzy Model Predictive Control of Discrete-Time Takagi–Sugeno Systems With Nonlinear Local Models," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2915–2925, 2018.

[61] P. H. Coutinho, R. F. Araújo, A.-T. Nguyen, and R. M. Palhares, "A Multiple-Parameterization Approach for local stabilization of constrained Takagi-Sugeno fuzzy systems with nonlinear consequents," *Information Sciences,* vol. 506, pp. 295–307, 2020.

[62] S. L. Chiu, "Fuzzy Model Identification Based on Cluster Estimation," *Journal of Intelligent and Fuzzy Systems*, vol. 2, no. 3, pp. 267–278, 1994.

[63] D. Gustafson and W. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, 1978.

[64] A. S. Mamaghani and W. Pedrycz, “Structural optimization of fuzzy rule-based models: Towards efficient complexity management,” *Expert Systems with Applications*, vol. 152, p. 113362, 2020.

[65] A. S. Mamaghani and W. Pedrycz, “Genetic-Programming based Architecture of Fuzzy Modeling: Towards Coping with High-dimensional Data,” *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2020.