PhD Thesis

KENESEI TAMÁS

DOI: 10.18136/PE.2014.563

PANNON EGYETEM

2014.

Számítási intelligencia alapú regressziós technikák és alkalmazásaik a folyamatmérnökségben

Értekezés doktori (PhD) fokozat elnyerése érdekében a
Pannon Egyetem Vegyészmérnöki Tudományok és Anyagtudományok
Doktori Iskolájához tartozóan

Írta:
Kenesei Tamás Péter

Konzulens: Dr. Abonyi János
Elfogadásra javaslom (igen/nem) ....
(aláírás)

A jelölt a doktori szigorlaton ..... %-ot ért el.

Az értekezés bírálóként elfogadásra javaslom:
Bíráló neve: igen/nem ....
(aláírás)
Bíráló neve: igen/nem ....
(aláírás)

A jelölt az értekezés nyilvános vitáján ....%-ot ért el.
Veszprém, ...
a Bíráló Bizottság elnöke

A doktori (PhD) oklevél minősítése ....

....
az EDT elnöke

Pannon Egyetem
Vegyészmérnöki és Folyamatmérnöki Intézet
Folyamatmérnöki Intézeti Tanszék

Számítási intelligencia alapú regressziós technikák és alkalmazásaik a folyamatmérnökségben

DOKTORI (PhD) ÉRTEKEZÉS

Kenesei Tamás

Konzulens
Dr. habil. Abonyi János, egyetemi tanár

Vegyészmérnöki és Anyagtudományok Doktori Iskola
2014.

University of Pannonia
Institute of Chemical and Process Engineering
Department of Process Engineering

Computational Intelligence based regression techniques and their applications in process engineering

PhD Thesis

Tamás Kenesei

Supervisor
János Abonyi, PhD, full professor

Doctoral School in Chemical Engineering and Material Sciences
2014.

Köszönetnyilvánítás

Mindazoknak akik hittek bennem.

"Együtt hajtunk, együtt halunk, rossz fiúk tűzön–vízen át!"

Kivonat

Számítási intelligencia alapú regressziós technikák és alkalmazásaik a folyamatmérnökségben

Az olyan adat alapú regressziós modellek mint a metsző hipersíkok, neurális hálózatok vagy szupport vektor gépek széles körben elterjedtek a szabályzásban, optimalizálásban és a folyamat monitorozásban. Mivel ezek a modellek nem értelmezhetőek, a folyamatmérnökök gyakran nem a legjobb gyakorlat szerint hasznosítják ezeket. Abban az esetben, ha betekintést nyerhetnénk ezekbe a fekete doboz modellekbe, lehetőségünk nyílna a modellek validálására, további információk és összefüggések feltárására a folyamat változok között, illetve a modell építés fázisát is tudnánk támogatni a-priori információk beépítésével.

Az értekezés kulcs gondolata, hogy a metsző hipersíkok, neurális hálózatok és szupport vektor gépek fuzzy modellekké alakíthatóak, és a kapott szabálybázis alapú rendszerek értelmezése biztosítható speciális modell redukciós és vizualizációs technikákkal.

Az értekezés első harmada a metsző hipersík alapú regressziós fák identifikációjával foglakozik. A működési tartomány rekurzívan particionált egy fuzzy c-regresszió alapú csoportosítási technikával. A kapott kompakt regressziós fa lokális lineáris modellekből áll. Ez a modellezési struktúra jól használható modell alapú szabályozásban, például modell prediktív szabályozás során.

A következő fejezet a neurális hálózatok validálásával, vizualizálásával és strukturális redukciójával foglalkozik, melyek alapjául a neurális hálózat rejtett rétegének fuzzy szabálybázissá történő átalakítása szolgál.

Végül a szupport vektor gépek és a fuzzy modellek közti analógia kerül betutatásra egy 3 lépéses redukciós algoritmus segítségével. A cél értelmezehető fuzzy regressziós modell, melynek alapja a szupport vektor regresszió.

A fejlesztett algoritmusok vegyészmérnöki gyakorlatban történő alkalmazhatóságát minden fejezetben esettanulmányok igazolják.

Abstract

Computational intelligence based regression techniques and their applications in process engineering

Data-driven regression models like hinging hyperplanes, neural networks and support vector machines are widely applied in control, optimization and process monitoring. Process engineers are often mistrustful of the application of these models since they are not interpretable. If we would have some insight to these black boxes we could have the possibility to validate these models, extract hidden information about the relationships among the process variables, and to support model identification by incorporating some prior knowledge.

The key idea of this thesis is that hinging hyperplanes, neural networks and support vector machines can be transformed into fuzzy models and the interpretability of the resulted rule-base systems can be ensured by special model reduction and visualization techniques.

The first part of the thesis deals with the identification of hinging hyperplane based regression trees. The operating regime of the model is recursively partitioned by a novel fuzzy c-regression clustering based technique. The resulted compact regression tree consists of local linear models, which model structure is favored in model based control solutions, like in model predictive control.

The next section deals with the validation, visualization and structural reduction of neural networks based on the transformation of the hidden layer of the network into an additive fuzzy rule base system.

Finally, based on the analogy of support vector regression and fuzzy models a three-step model reduction algorithm will be proposed to get interpretable fuzzy regression models on the basis of support vector regression.

Real life utilization of the developed algorithms is shown by sectionwise examples taken from the area of process engineering.

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Network
APC	Advanced process control
CSTR	Continuous stirred tank reactor
FIS	Fuzzy system
FCRM	Fuzzy c–regression clustering method
DT	Decision tree
FAS	Fuzzy additive system
LOLIMOT	Locally linear model tree
KDD	Knowledge data discovery in databases
HH	Hinging hyperplanes
PDFS	Positive definite fuzzy system
SVM	Support vector machine
SVR	Support vector regression
NARX	Nonlinear Autoregressive with eXogenous Input
NN	Neural Network
MDS	Multi-dimensional scaling
MLP	Multilayer perceptron
MPC	Model predictive control
RBF	Radial basis function
RS	Reduced set method
RMSE	Root main square error
OLS	Orthogonal least squares
TS	Takagi-Sugeno fuzzy model

Notations

Regression problem

x	independent variable
y	dependent variable
θ	model parameter
ρ	cardinality

Fuzzy logic

δ	rule consequent
$\mu_{i,k}$	membership value, where i stands for the clusters and k for the data points ($k = 1 \dots n$).
Λ	a priori constraints
$\mathbf{v}$	cluster center
$\mathbf{U}$	fuzzy partition matrix

Dynamic systems

E, J	cost function
n_a	output order
n_b	input order
α	filter value
$Q(u)$	cartridge heater performance
Q_M	maximum power
u	heating signal
T_{out}	outlet temperature
H_p	prediction horizon
H_c	control horizon
λ	weighting coefficient
α	filter parameter

Neural networks and support vector machines

β	firing strenght
τ	bias
ϕ	feature mapping
S	similarity measure
k	kernel function
$*$	*interactive or* operator
ε	error tolerance
ξ_i, ξ_i^*	slack variables
C	cost coefficient
N_i	input space dimension
N_x	number of support vectors
N_R	reduced set expansion

Chapter 1

Introduction

With combination of computation intelligence tools, like hinging hyperplanes (HH), support vector regression (SVR), artificial neural networks (ANNs) and fuzzy models powerful and interpretable models can be developed. This introduction presents the motivation of handling these techniques in one integrated framework and describes the structure of the thesis.

1.1 Data-driven techniques in process engineering

Information for process modeling and identification can be obtained from different sources. According to the type of available information, three basic levels of model synthesis are defined:

White-box or first-principle modeling A complete mechanistic model is constructed from a priori knowledge and physical insight. Here dynamic models are derived based on mass, energy and momentum balances of the process [1].

Fuzzy logic modeling A linguistically interpretable rule-based model is formed based on the available expert knowledge and measured data [1].

Black-box model or empirical model No physical insight is available or used, but the chosen model structure belongs to families that are known to have good flexibility and have been "successful in the past". Model parameters are identified based on measurement data [1].

This means, if we have good mechanistic knowledge about the process, this can be transformed into white box model described by analytical (differential) equations. If we have information like human experience described by linguistic rules and variables, the mechanistic modeling approach is useless and the application of rule-based approaches like fuzzy logic is more appropriate [2, 3]. Finally, there may be situations, where the most valuable information comes from input-output data collected during operation. In this case, the application of black box models is the best choice. These black box models are especially valuable, when an accurate model of the process dynamics is needed. Therefore, the nonlinear black box modeling is a challenging and promising research field [4, 5, 6, 7, 8].

Black-box models are especially valuable when an accurate model of the process dynamics is needed. In order to perform a successful data-driven model the following steps have to be carried out [1]:

1. Selection of model structure and complexity
2. Design of exctitation signals used or identification
3. Identification of model parameters
4. Model validation

Process engineers are often mistrustful of the application of nonlinear black box models since they are not interpretable. If we would have some insight to these black boxes we could have the possibility to validate these models, extract hidden information about the relationships among the process variables, selection of the model structure based on this knowledge, and to support model identification by incorporating some prior knowledge.

For these purposes novel model identification methods, interpretable, robust and transparent models are needed. Since we are interested in extraction of knowledge from process data, tools and methodologies of data mining should be also efficiently utilized.

Historically the notion of finding useful patterns in data has been given a variety of names including data mining, knowledge extraction, information discovery, and data pattern processing. The term data mining has been mostly used by statisticians, data analysts, and the management information systems (MIS) communities [9]. Data mining is not just a simple tool, but a complex process consisting of multiple steps, hence this process must be integrated into the supported activity. The process of data–based knowledge discovery can be seen on Fig. 1.1. Introducing the knowledge discovery process allows us to give a brief definition to data mining:

Data mining is a decision support process to give valid, useful and a priori not known reliable information from data sources. [10]

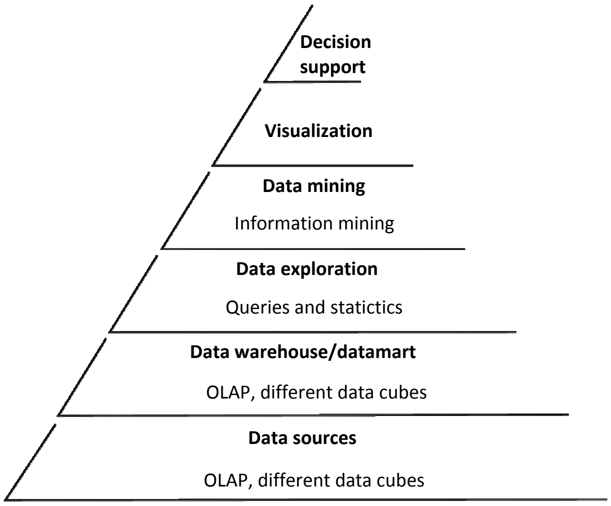

Figure 1.1: Steps of the knowledge discovery process

To understand this definition the following keywords must be further investigated [1]:

process Data mining is not a product–ready delivered software generating automatically consumable knowledge from stored data, but it is a complex process consisting of well–defined steps. Regression techniques takes place during model preparation. Improved model identification algorithms and interpretable models ensure quality of acquired information at the end of KDD process.

valid Mined information must be accurate and statistically significant. Validity states not only accuracy but also completeness.

useful It is not enough to generate valid knowledge with the help of data mining, explored knowledge must be utilizable for the exactly defined problem. Unfortunately measuring usefulness is not always solved, as sometimes affect of the used information cannot be measured with monetary tools.

preliminarily not known Strictly speaking, data–based knowledge discovery has twofold aim: confirmation and discovery. Confirmation means strengthening hypothesis of the data expert while discovery stands for identification of patterns generated by the examined system. Aim of data mining basically is to identify discoverable knowledge to define predictive or descriptive functions. Regression–based methods are predictive exercises defining future, not known properties or behaviors of the modeled process.

exact Result of data mining must be easily interpretable, and the model should not deviate from from reality.

Key challenge of data mining is to capture potential information from opaque data sources and transform data to a more compact, abstract, informative and easy-to-use way. Hence, data mining looks for trends and patterns in large databases. Knowledge delivered by data mining exists in a form of an interpretable model or information represented in decision trees, rule bases, networks or mathematical equations.

The aim of the thesis at hand is to extract interpretable regression models to foster the usage of these models in process engineering.

1.2 Interpretability and model structure identification

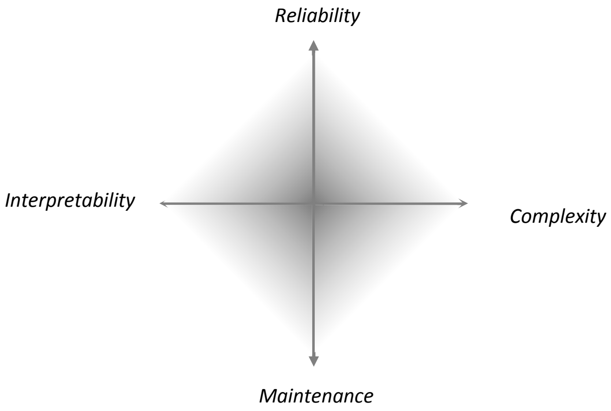

Figure 1.2: Tradeoffs in modeling

Zadeh stated in Principle of Incompatibility [11] "*as the complexity of a system increases our ability to make precise and yet significant statements about its behavior diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics.*" Obtaining high degree of interpretability with sizeable complexity is a contradictory purpose and - in practice - one of these properties prevails over the other. The problem becomes much more difficult to handle when model reliability and maintenance is included to the conditions.

In most studies of process identification it is assumed that there is an optimal functional structure describing relationship between measured input and output data. It is very difficult to find a functional structure for a nonlinear system. Generally speaking, the structure identification of a system has to solve two problems: one is to find input variables and one is to find input-output relations. The selection the input variables can be based on the aim of the modeling exercise and on prior knowledge related to the system to be modeled. For static systems statistical techniques, correlation analysis and modeling performance based methods proposed by Sugeno [12, 13, 14] and Jang [15] can be used. For dynamical systems, the selection of the relevant model-inputs is identical to the determination of the model order of the NARX model.

The identified model must be validated as well. If the model is validated by the same data set from which it was estimated, the accuracy of the model always improves as the complexity of the model structure increases. In practice, a trade-off is usually sought between model accuracy and complexity, and there are several approaches to compensate for this automatic increase of the modeling performance.

Fig. 1.3 shows connection between modeling error and model complexity[1]. As this figure shows, selection of the proper model structure is a complex tasks that requires careful selection of training and validation datasets, proper cost functions and proper optimization strategies or heuristics that support the modeler.

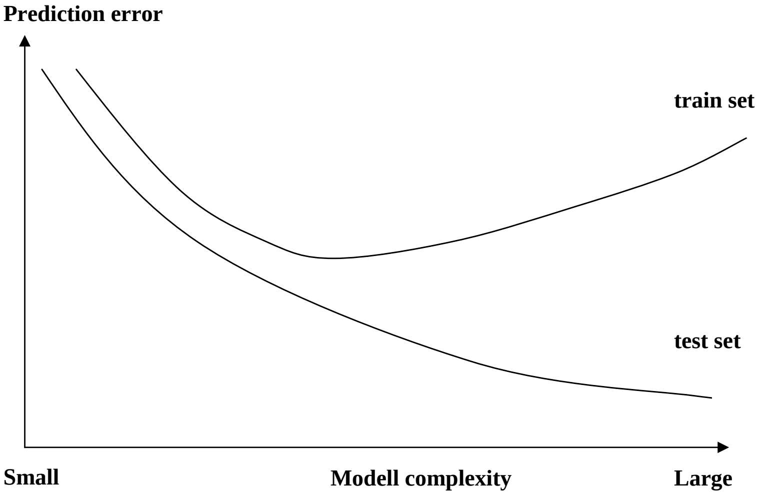

Figure 1.3: Model complexity versus modeling error

1.3 Computational intelligence based models

Majority of problems arose in process engineering practice requires data-driven modeling of nonlinear relationships between experimental and technological variables. Complexity of nonlinear regression techniques is gradually expanding with the development of analytical and experimental techniques, hence model structure and parameter identification is a current and important topic in the field of nonlinear regression not just by scientific but also from industrial point of view as well.

In line with these expectations and taking interpretability of regression models as basic requirement aim of this thesis is the development of robust computational intelligence models in order to solve nonlinear regression identification tasks.

Tools from the armory of computational intelligence (also referred as soft computing) have been in focus of researches recently, since soft computing techniques are used for fault detection, forecasting of time-series data, inference, hypothesis testing, and modeling of causal relationships (regression techniques) in process engineering.

The meaning of soft computing was originally tailored in the early 1990s by Dr. Zadeh [16]. Soft computing refers to a collection of computational techniques in computer science, artificial intelligence, machine learning and some engineering disciplines, to solve two cardinal problems:

- Learning from experimental data (examples, samples, measurements, records, patterns) by neural networks and support vector based techniques

- Embedding existing structured human knowledge(experience, expertise, heuristic) into fuzzy models [17]

These approaches attempt to study, model, and analyze very complex phenomena: those for which more conventional methods have not yielded low cost, analytic, and complete solutions. Earlier computational approaches (hard computing) could model and precisely analyze only relatively simple systems.

As more complex systems arising in biology, medicine, the humanities, management sciences, and similar fields often remained intractable to conventional mathematical and analytical methods. Where hard computing schemes –striving for exactness and full truth–fail to render the given problem, soft computing techniques deliver robust, efficient and optimal solutions to capture available design knowledge for further analysis.

Generally speaking, soft computing techniques resemble biological processes more closely than traditional techniques, which are largely based on formal logical systems, such as sentential logic and predicate logic, or rely heavily on computer-aided numerical analysis. Hence in real life high degree of uncertainty should be taken in to account during identification process. Soft computing tries to solve this challenge with exploiting tolerance for imprecision, uncertainty and partial truth in order to reach robustness and transparency at low cost.

Many systems are not amenable to conventional modeling approaches due to the lack of precise, formal knowledge about the system, due to strongly nonlinear behavior, high degree of uncertainty, or time-varying characteristics. Computational intelligence, the technical umbrella of hinging hyperplanes (HH) [18, 19, 20], support vector regression (SVR) [21, 22, 23, 24, 25], artificial neural networks (ANNs) [26, 27, 28, 29, 30, 31] and fuzzy logic [2, 3, 32] has been recognized as a powerful tool which is tolerant of imprecision and uncertainty, and can facilitate the effective development of models by combining information from different sources, such as first-principle models, heuristics and data.

Table 1.1: Evaluation criteria system

Property	**Description**
Interpolation behavior	Character of the model output between training data samples
Extrapolation behavior	Character of the model outside region of training data
Locality	Locality, globality of the basis functions
Accuracy	Model accuracy with given number of parameters
Smoothness	Smoothness of model output
Sensitivity to noise	Affect of noise on model behavior
Parameter optimization	Can linear and nonlinear model parameters estimated
Structure optimization	Possibilities of model structure and complexity optimization
Online adaptation	Possibilities of on-line model adaptable
Training speed	Speed of model parameter estimation
Evaluation speed	Speed of model evaluation
Curse of dimensionality	Model scale up to higher input space dimensions
Interpretation	Interpretation of model parameters and model structure
Incorporation of constraints	Difficulty of constraint incorporation
Usage	Acceptability and penetration of modeling structure

Among the techniques of computational intelligence, ANNs attempt to mimic the structures and processes of biological neural systems. They provide powerful analysis properties such as complex processing of large input/output information arrays, representing complicated nonlinear associations among data, and the ability to generalize or form concepts-theory. Support vector regression in it's nature is very similar to ANNs and on the other hand HH models can be a good alternative to NNs.

Based on [1] Table 1.1 summarizes several criterion can be used to evaluate these modeling techniques. The studied nonlinear regression techniques have usually robust modeling structure, however the resulted model is often a non–interpretable black–box model. This thesis focuses on the identification, utilization and interpretability of ANNs, HHs and SVR in the realm of modeling, identification and control of nonlinear processes.

1.4 Motivation and outline of the thesis

This thesis has twofold aims to introduce new algorithms for nonlinear regression (hinging hyperplanes) and to highlight possibilities to transform black–box non-linear regression models (neural networks and support vector regression) to transparent and interpretable fuzzy rule base (see fig 1.4). These techniques together form a framework to utilize combination-of-tools-methods in order to understand, visualize and validate non–linear black box models.

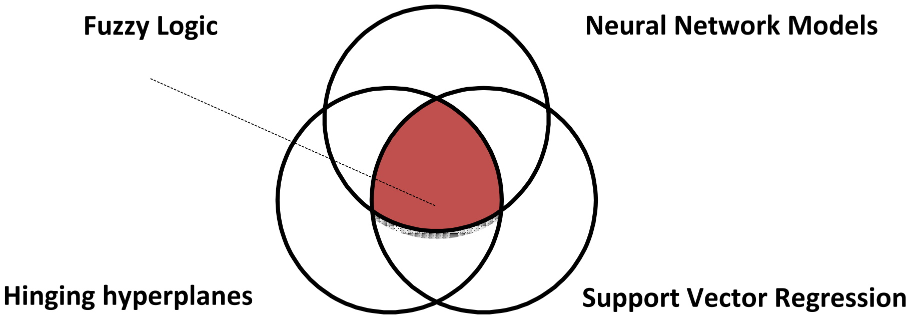

Figure 1.4: Framework of the thesis

Three algorithms were examined based on the evaluation criteria system mentioned in section 1.2 in details namely identification of regression trees based hinging hyperplane, neural networks and support vector regression. Application of these techniques eventuate black box models at first step. It will be shown how interpretability could be maintained during model identification with utilization of applicable visualization and model structure reduction techniques within the fuzzy modeling framework.

Chapter 2 Hinging hyperplanes deals with the identification of hinging hyperplane based regression trees. Results of the developed algorithm proves that the implementation of a priori constraints enables fuzzy c-regression clustering technique to identify hinging hyperplane models. Application of this technique recursively on the partitioned input space ends up in a regression tree capable for modeling and even for implementation of model predictive control of technological data coming from real life applications. According to the evaluation system mentioned in section 1.2 the development algorithm contains major developments in hinge model identification, since the proposed method has higher *accuracy*, the tree–based representation enables better *structure* and *parameter optimization* and helps *interpretation* of model parameters and model structure.

Main improvements of hinging hyperplane identification is discussed in
T. Kenesei, B. Feil, J. Abonyi, Fuzzy Clustering for the Identification of Hinging Hyperplanes Based Regression *Lecture notes in computer science, Lecture notes in artificial intelligence; 4578. ISBN:9783540733997* pp. 179-186. 2007

Detailed description of regression tree representation can be found in
T. Kenesei, J. Abonyi, Hinging hyperplane based Regression tree identified by Fuzzy Clustering *WSC16 - 16th Online World Conference on Soft Computing in Industrial Applications* 2011.

Hinging hyperplane model predictive control is published in
T. Kenesei, B. Feil, J. Abonyi, Identification of Dynamic Systems by Hinging Hyperplane Models *ICAI 2007 - 7th International Conference on Applied Informatics Eger 2007.*

Our efforts within the framework of hinging hyperplane identification and control is submitted to
T. Kenesei, J. Abonyi, Hinging hyperplane based Regression tree identified by Fuzzy Clustering and its application *Applied Soft Computing Journal* vol 13(2) pp. 782-792 2013.

Chapter 3 Visualization and reduction of neural networks deals with the validation, visualization and structural reduction of neural networks in order to achieve better *interpretability*. With applying orthogonal least squares and similarity measure based techniques *structure optimization* is also performed. It is described in details that the hidden layer of the neural network can be transformed to an additive fuzzy rule base.
Reduction and visualization methods are described in
T. Kenesei, B. Feil, J. Abonyi, Visualization and Complexity Reduction of Neural Networks *Applications of soft computing: updating the state of art.*, pp. 43-52. 2009. *Advances in soft computing ISBN:9783540880783* vol. 52.

Chapter 4 Interpretable Support vector regression describes connections between fuzzy regression and support vector regression, and introduces a three-step reduction algorithm to get interpretable fuzzy regression models on the basis of support vector regression. This combination–of–tools technique retains good *general-*

ization behavior and noise *insensitiveness* of the support vector regression however keeps the identified model *interpretable* and with the application of reduction techniques *structure optimization* is also achieved.

Three-step reduction algorithm and visualization methods can be found in
T. Kenesei, A. Roubos, J. Abonyi, A Combination-of-Tools Method for Learning Interpretable Fuzzy Rule-Based Classifiers from Support Vector Machines *Lecture Notes in Computer Science; 4881. ISBN:978-3-540-77225-5* pp. 477-486. 2008

Application of support vector regression models is described in the following publications
T. Kenesei, J. Abonyi, Interpretable Support Vector Machines in Regression and Classification- Application in Process Engineering, *Hungarian Journal of Industrial Chemistry*, VOL 35. pp. 101-108 2007.
T. Kenesei, J. Abonyi, Interpretable Support Vector Regression, *Artificial Intelligence Research*, Vol 1 (2), ISSN:1927-6974 ,2012

Real life utilization of the developed algorithms is shown by section–wise examples taken from the area of chemical engineering. Finally, Chapter 5 summarizes the new scientific results in English and in Hungarian.

The proposed framework supports the structure and parameter identification of regression models. To achieve this goal structure optimization techniques like orthogonal least squares (OLS) and decision tree based model representations are applied. The detailed description of the regression problem are given in Appendix A while OLS is described in details in Appendix C.

As can be seen, this thesis is based on a number of papers we published recently. I have attempted to eliminate redundancy of these papers. To promote easier reading consistent nomenclature list can be found in the Notations section. Source codes of the utilized softwares written in Matlab can be found on the www.abonyilab.com website.

Chapter 2

Hinging Hyperplanes

Hinging hyperplane model is proposed by Breiman [20] and identification of this type of non-linear model is several times reported in the literature due to suffering from convergency and range problems [33, 34, 35, 19]. Methods like penalty of hinging angle were proposed to improve Breiman's algorithm [18], or Gauss–Newton algorithm can be used to obtain the final non–linear model [34]. Several application examples have been also published in the literature, e.g. it can be used in identification of piecewise affine systems via mixed-integer programming [36] and this model also lends himself to form hierarchical models [19].

In this chapter a much more applicable algorithm is proposed for hinging hyperplane identification. The key idea is that in a special case ($c = 2$) fuzzy c-regression method (FCRM) [37] can be used for identifying hinging hyperplane models. To ensure that two local linear models used by fuzzy c-regression algorithm form a hinging hyperplane function, it has to be granted that local models are intersecting each other in the operating regime of the model. The proposed constrained FCRM algorithm is able to identify one hinging hyperplane model, therefore to generate more complex regression trees, described method should be recursively applied. Hinging hyperplane models containing two linear submodels divide operating region of the model into two parts, since hinging hyperplane functions define a linear separating function in the input space of the hinging hyperplane function. Sequence of these separations result a regression tree where branches correspond to linear division of operating regime based on the hinge of the hyperplanes at a given node. This type of partitioning can be considered as crisp version of a fuzzy regression based tree described in [38]. Fortunately, in case of hinging hyperplane based regression tree there is no need for selecting best splitting variable at a given node, but on the other hand it is not as interpretable as regression trees utilizing univariate decisions at nodes.

The proposed modeling framework is based on the algorithm presented at the 16th Online Conference on Soft Computing in Industrial Applications [39]. To support the analysis and building of this special model structure novel model performance and complexity measures are presented in this work. Special attention is given for the modeling and controlling nonlinear dynamical systems. Therefore, application example related to Box–Jenkins gas furnace benchmark identification problem is added. It will be also shown that thanks to the piecewise linear model structure the resulted regression tree can be easily utilized in model predictive control. A detailed application example related to the model predictive control of a water heater will demonstrate the benefits of the proposed framework.

A critical step in the application of model-based control is the development of a suitable model for the process dynamics. This difficulty stems from lack of knowledge or understanding of the process to be controlled. Fuzzy modeling has been proven to be effective for the approximation of uncertain nonlinear processes. Recently, nonlinear black-box techniques using fuzzy and neuro-fuzzy modeling have received a great deal of attention [40]. Readers interested in industrial applications can find an excellent overview in [41]. Details of model-based control relevant applications are well presented in [42] and [43].

Most nonlinear identification methods are based on the NARX (Nonlinear AutoRegressive with eXogenous input) model [8]. The use of NARX black box models for high-order dynamic processes in same cases are impractical. Data–driven identification techniques alone, may yield unrealistic NARX models in terms of steady-state characteristics, local behavior and unreliable parameter values. Moreover, the identified model can exhibit regimes which are not found in the original system [43]. This is typically due to insufficient information content of the identification data and the over-parametrization of the model. This problem can be remedied by incorporating prior knowledge into the identification method by constraining the parameters of the model [44]. Another possibility to reduce the effects of over-parametrization is to restrict the structure of the NARX model, using for instance the Nonlinear Additive AutoRegressive with eXogenous input (NAARX) model [45]. In this thesis a different approach is proposed, a hierarchial set of local linear models are identified to handle complex systems dynamics.

Operating regime based modeling is a widely applied technique for identification of these nonlinear systems. There are two approaches for building operating regime based models. An additive model uses sum of certain basis functions to represent a non-linear system, while partitioning approach partitions the input space recursively to increase modeling accuracy locally [18]. Models generated by this approach are often represented by trees [46]. Piecewise linear systems [47] can be easily represented in a regression tree structure [48]. Special type of regression tree is called locally linear model tree, which algorithm combines a heuristic strategy for input space decomposition with a local linear least squares optimization (like LOLIMOT [1]). These models are hierarchical models consisting of nodes and branches. Internal nodes represent tests on input variables of the model, and branches correspond to outcomes of said tests. Leaf (terminal) nodes contains regression models in case of regression trees.

Thanks to the structured representation of the local linear models, hinging hyperplanes lend themselves to a straightforward incorporation in model based control schemes. In this chapter this beneficial property is demonstrated in the design of instantaneous linearization based model predictive control algorithm [32].

This chapter organized as follows: next section discusses how hinging hyperplane functions' approximation is done with FCRM identification approach. The description of tree growing algorithm and the measures proposed to support model building are given in Section 2.2. In Section 4.3, application examples are presented while Section 4.4 concludes the chapter.

2.1 Identification of hinging hyperplanes

2.1.1 Hinging hyperplanes

The following section gives a brief description about the hinging hyperplane approach on the basis of [34, 18, 49], followed by how the constrains can be incorporated into FCRM clustering.

For a sufficiently smooth function $f(\mathbf{x}_k)$, which can be linear or non-linear, assuming that regression data $\{\mathbf{x}_k, y_k\}$ is available for $k = 1, \dots, N$. Function $f(\mathbf{x}_k)$ can be represented as the sum of a series of hinging hyperplane functions $h_i(\mathbf{x}_k), i = 1, 2, \dots, K$ are defined as the hinging hyperplane function. Breiman[20] proved that we can use hinging hyperplane to approximate continuous functions on compact sets, guaranteeing a bounded approximation error

$$\|e_n\| = \|f - \sum_{i=1}^{K} h_i(\mathbf{x})\| \leq (2R)^4 c^2/K \tag{2.1}$$

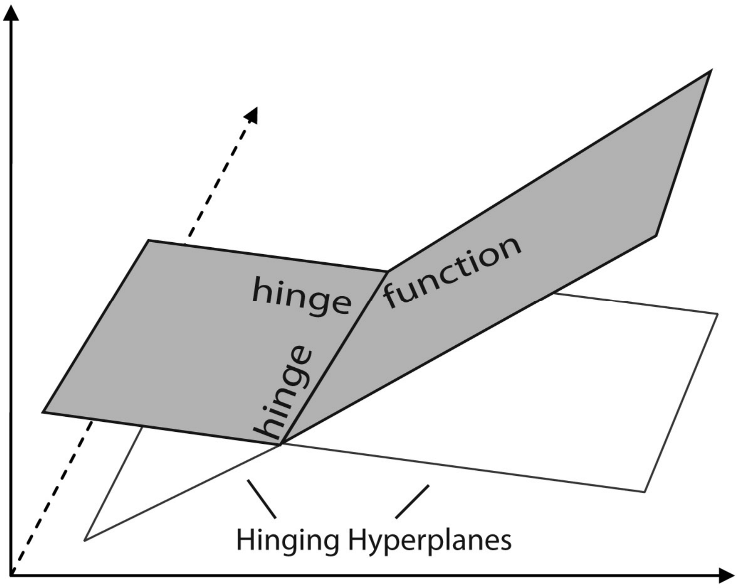

Figure 2.1: Basic hinging hyperplane definitions

where K is the number of hinging hyperplane functions R is the radius of the sphere in which the compact set is contained and c is such that

$$\int \|w\|^2 |f\,(w)\,| dw = c < \infty \tag{2.2}$$

The approximation with hinging hyperplane functions can get arbitrarily close if sufficiently large number of hinging hyperplane functions are used. The sum of the hinging hyperplane functions $\sum_{i=1}^{K} h_i(\mathbf{x}_k)$ constitutes a continuous piecewise linear function. The number of input variables n in each hinging hyperplane function and the number in hinging hyperlane functions K are two variables to be determined. The explicit form for representing a function $f(\mathbf{x}_k)$ with hinging hyperplane functions becomes (see Fig. 2.1)

$$f(\mathbf{x}_k) = \sum_{i=1}^{K} h_i(\mathbf{x}_k) = \sum_{i=1}^{K} \langle \max \,|\, \min \rangle \left(\mathbf{x}_k^T \theta_{1,i}, \mathbf{x}_k^T \theta_{2,i}\right) \tag{2.3}$$

where $\langle \max \,|\, \min \rangle$ means max or min.

Suppose two hyperplanes are given by:

$$y_k = \mathbf{x}_k^T \theta_1, y_k = \mathbf{x}_k^T \theta_2 \tag{2.4}$$

where $\mathbf{x}_k = [x_{k,0}, x_{k,1}, x_{k,2}, \ldots, x_{k,n}]$, $x_{k,0} \equiv 1$ is the kth regressor vector and y_k

is the kth output variable. These two hyperplanes are continuously joined together at $\{\mathbf{x} : \mathbf{x}^T(\theta_1 - \theta_2) = 0\}$ as can be seen in Fig. 2.1. As a result they are called *hinging hyperplanes*. The joint $\triangle = \theta_1 - \theta_2$, multiples of $\triangle$ are defined *hinge* for the two hyperplanes, $y_k = \mathbf{x}_k^T\theta_1$ and $y_k = \mathbf{x}_k^T\theta_2$. The solid/shaded part of the two hyperplanes explicitly given by

$$y_k = \max(\mathbf{x}_k^T\theta_1, \mathbf{x}_k^T\theta_2) \text{ or } y_k = \min(\mathbf{x}_k^T\theta_1, \mathbf{x}_k^T\theta_2) \tag{2.5}$$

Hinging hyperplane method has some interesting advantages for non-linear function approximation and identification:

1. Hinging hyperplanes functions could be located by a simple computationally efficient method. In fact hinging hyperplane models are piecewise linear models, the linear models are usually obtained by repeated use of linear least-squares method, which is very efficient. The aim is to improve the whole identification method with more sophisticated ideas.

2. For non–linear functions with resemble hinging hyperplane functions, the hinging hyperplane method has very good and fast convergence properties.

Hinging hyperplane method practically combines some advantages of neural networks (in particular ability to handle very large dimensional inputs) and of constructive wavelet based estimators (availability of very fast training algorithms).

Essential hinging hyperplane search problem can be viewed as an extension of linear least-squares regression problem. Linear least-squares regression aims to find the best parameter vector $\widehat{\theta}$, by minimizing a quadratic cost function with which regression model gives the best linear approximation to y. For nonsingular data matrix $\mathbf{X}$ linear least squares estimate $y = \mathbf{x}^T\theta$ is always uniquely available. The hinging hyperplane search problem, on the other hand, aims to find the two parameter vectors θ_1 and θ_2, defined by

$$[\theta_1, \theta_2] = \arg\min_{\theta_1,\theta_2} \sum_{k=1}^{N} \left[\langle \max | \min \rangle \left(y_k - \mathbf{x}_k^T\theta_1, y_k - \mathbf{x}_k^T\theta_2\right)\right]^2 \tag{2.6}$$

A brute force application of Gauss-Newton method can solve the above described optimization problem. However, two problems exist [18]:

1. High computational requirement. The Gauss–Newton method is computationally intensive. In addition, since the cost function is not continuously

differentiable, the gradients required by Gauss-Newton method can not be given analytically. Numerical evaluation is thus needed which has high computational demand.

2. Local minima. There is no guarantee that the global minimum can be obtained. Therefore appropriate initial condition is crucial.

2.1.2 Improvements in hinging hyperplane identification

The proposed identification algorithm applies a much simpler optimization method, the so–called alternating optimization which is a heuristic optimization technique and has been applied for several decades for many purposes, therefore it is an exhaustively tested method in non–linear parameter and structure identification as well. Within the hinging hyperplane function approximation approach, the two linear submodels can be identified by the weighted linear least-squares approach, but their operating regimes (where they are valid) are still an open question.

For that purpose fuzzy c-regression model (further referred as FRCM and proposed by Hathaway and Bezdek [37]) was used. This technique is able to partition the data and determine the parameters of the linear submodels simultaneously. With the application of alternating optimization technique and taking advantage of the linearity in $(y_k - \mathbf{x}_k^T\theta_1)$ and $(y_k - \mathbf{x}_k^T\theta_2)$, an effective approach is given for hinging hyperplane function identification, hence FCRM method in a special case ($c = 2$) is able to identify hinging hyperplanes. The proposed procedure is attractive in local minima point of view as well, because in this way although the problem is not avoided but transformed into a deeply discussed problem, namely the cluster validity problem.

The following quadratic cost function can be applied for the FCRM method

$$E_m(\mathbf{U}, \{\theta_i\}) = \sum_{i=1}^{c}\sum_{k=1}^{N}(\mu_{i,k})^m E_{i,k}(\theta_i) \tag{2.7}$$

where $m \in \langle 1, \infty)$ denotes a weighting exponent which determines the fuzziness of the resulting clusters, while θ_i represents the parameters of local models and $\mu_{i,k} \in \mathbf{U}$ is the membership degree, which could be interpreted as a weight representing the extent to which the value predicted by the model $f_i(\mathbf{x}_k, \theta_i)$ matches y_k. The prediction error is defined by:

$$E_{i,k} = \big(y_k - f_i(\mathbf{x}_k;\theta_i)\big)^2 \tag{2.8}$$

but other measures can be applied as well, provided they fulfill the minimizer property stated by Hathaway and Bezdek [37].

One possible approach to the minimization of the objective function (2.7) is the group coordinate minimization method that results in the following algorithm:

- **Initialization** Given a set of data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ specify c, the structure of the regression models (2.8) and the error measure (2.7). Choose a weighting exponent $m > 1$ and a termination tolerance $\epsilon > 0$. Initialize the partition matrix randomly.

- **Repeat** For $l = 1, 2, \ldots$

Step 1 Calculate values for the model parameters θ_i that minimize the cost function $E_m(\mathbf{U}, \{\theta_i\})$.

Step 2 Update the partition matrix

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c}(E_{i,k}/E_{j,k})^{2/(m-1)}},\ 1 \le i \le c,\ 1 \le k \le N \tag{2.9}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

A specific situation arises when the regression functions f_i are linear in the parameters θ_i, $f_i(\mathbf{x}_k;\theta_i) = \mathbf{x}_{i,k}^T\theta_i$, where $\mathbf{x}_{i,k}$ is a known arbitrary function of $\mathbf{x}_k$. In this case, the parameters can be obtained as a solution of a set of weighted least-squares problem where the membership degrees of the fuzzy partition matrix $\mathbf{U}$ serve as the weights.

The N data pairs and the membership degrees are arranged in the following matrices.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{i,1}^T \\ \mathbf{x}_{i,2}^T \\ \vdots \\ \mathbf{x}_{i,N}^T \end{bmatrix},\ \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix},\ \mathbf{\Phi}_i = \begin{bmatrix} \mu_{i,1} & 0 & \cdots & 0 \\ 0 & \mu_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_{i,N} \end{bmatrix} \tag{2.10}$$

The optimal parameters θ_i are then computed by:

$$\theta_i = [\mathbf{X}^T \mathbf{\Phi}_i \mathbf{X}]^{-1} \mathbf{X}^T \mathbf{\Phi}_i \mathbf{y} \tag{2.11}$$

Applying $c = 2$ during FCRM identification these models can be used as base identifiers for hinging hyperplane functions. For hinging hyperplane function identification purposes, two prototypes have to be used by FCRM ($c = 2$), and these prototypes must be linear regression models. However, these linear submodels have to intersect each other within the operating regime covered by the known data points (within the hypercube expanded by the data). This is a crucial problem in the hinging hyperplane identification area [18]. To take into account this point of view as well, constrains have to be taken into consideration as follows. Cluster centers $\mathbf{v}_i$ can also be computed from the result of FCRM as the weighted average of the known input data points

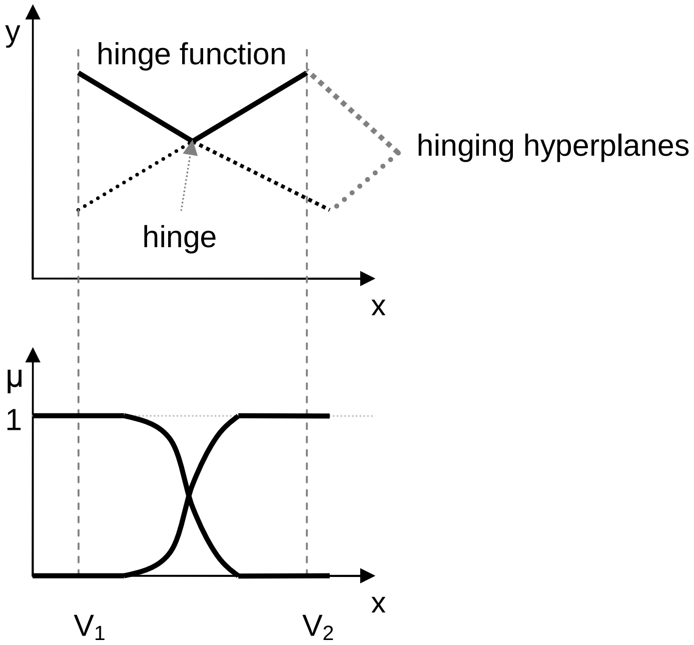

Figure 2.2: Hinging hyperplane identification restrictions

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} \mathbf{x}_k \mu_{i,k}}{\sum_{k=1}^{N} \mu_{i,k}} \tag{2.12}$$

where the membership degree $\mu_{i,k}$ is interpreted as a weight representing the extent to which the value predicted by the model matches y_k. These cluster centers are located in the 'middle' of the operating regime of the two linear submodels. Because the two hyperplanes must cross each other following criteria can be specified (see

Fig. 2.9):

$$\begin{aligned} &\mathbf{v}_1(\theta_1-\theta_2)<0 \quad \text{and} \quad \mathbf{v}_2(\theta_1-\theta_2)>0 \text{ or} \\ &\mathbf{v}_1(\theta_1-\theta_2)>0 \quad \text{and} \quad \mathbf{v}_2(\theta_1-\theta_2)<0 \end{aligned} \tag{2.13}$$

These relative constrains can be used to take into account the constrains above:

$$\mathbf{\Lambda}_{rel,1,2}\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\leq 0 \text{ where } \mathbf{\Lambda}_{rel,1,2}=\begin{bmatrix}\mathbf{v}_1 & -\mathbf{v}_1\\ -\mathbf{v}_2 & \mathbf{v}_2\end{bmatrix} \tag{2.14}$$

When linear equality and inequality constraints are defined on these prototypes, quadratic programming (QP) has to be used instead of the least-squares method. This optimization problem still can be solved effectively compared to other constrained nonlinear optimization algorithms.

Local linear constraints applied to fuzzy models can be grouped into the following categories according to their validity region:

- **Local constrains** are valid only for the parameters of a regression model, $\mathbf{\Lambda}_i\theta_i \leq \omega_i$.
- **Global constrains** are related to all of the regression models, $\mathbf{\Lambda}_{gl}\theta_i \leq \omega_{gl}$, $i = 1, \ldots, c$.
- **Relative constrains** define the relative magnitude of the parameters of two or more regression models.

$$\mathbf{\Lambda}_{rel,i,j}\begin{bmatrix}\theta_i\\ \theta_j\end{bmatrix}\leq \omega_{rel,i,j} \tag{2.15}$$

An example for these types of constrains are illustrated in Fig.2.3.

In order to handle relative constraints, the set of weighted optimization problems has to be solved simultaneously. Hence, the constrained optimization problem is formulated as follows:

$$\min_{\theta}\left\{\frac{1}{2}\theta^T\mathbf{H}\theta+\mathbf{c}^T\theta\right\} \tag{2.16}$$

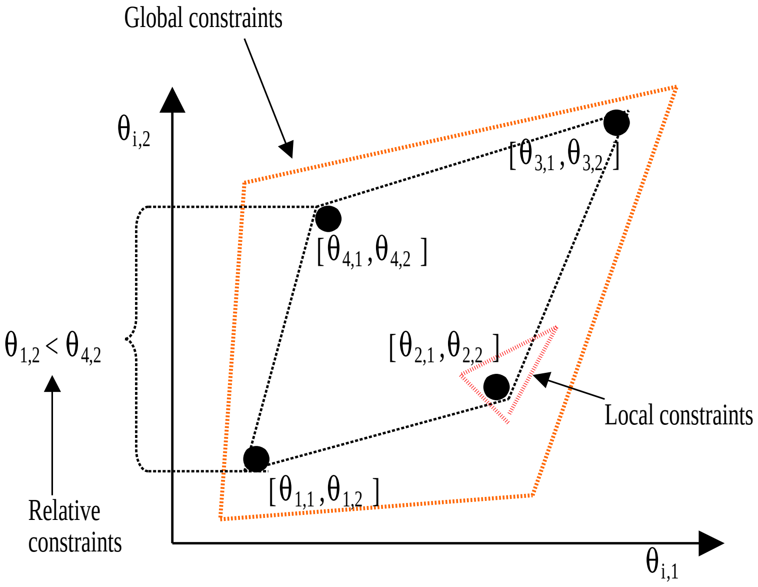

Figure 2.3: Hinging hyperplane model with 4 local constraints and two parameters

with $\mathbf{H} = 2\mathbf{X}'^T\mathbf{\Phi}\mathbf{X}'$, $\mathbf{c} = -2\mathbf{X}'^T\mathbf{\Phi}\mathbf{y}'$, where

$$\mathbf{y}' = \begin{bmatrix} \mathbf{y} \\ \mathbf{y} \\ \vdots \\ \mathbf{y} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_c \end{bmatrix}, \tag{2.17}$$

$$\mathbf{X}' = \begin{bmatrix} \mathbf{X}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{X}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X}_c \end{bmatrix}, \quad \mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{\Phi}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Phi}_c \end{bmatrix} \tag{2.18}$$

where $\mathbf{\Phi}_i$ contains local membership values and the constraints on θ:

$$\mathbf{\Lambda}\theta \leq \omega \tag{2.19}$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{\Lambda}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Lambda}_c \\ \mathbf{\Lambda}_{gl} & 0 & \cdots & 0 \\ 0 & \mathbf{\Lambda}_{gl} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Lambda}_{gl} \\ \{\mathbf{\Lambda}_{rel}\} & & & \end{bmatrix}, \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_c \\ \omega_{gl} \\ \omega_{gl} \\ \vdots \\ \omega_{gl} \\ \{\omega_{rel}\} \end{bmatrix}. \tag{2.20}$$

Referring back to Fig.2.1 it can be concluded with this method both part of the intersected hyperplanes are described and that part ($\langle max|min \rangle$) is selected which describes the the training data in the most accurate way.

2.2 Hinging hyperplane based binary trees

So far, the hinging hyperplane function identification method is presented. The proposed technique can be used to determine the parameters of one hinging hyperplane function. The classical hinging hyperplane approach can be interpreted by identifying K hinging hyperplane models consisting of global model pairs, since these operating regimes cover the whole N dataset. This representation leads to several problems not just during model identification but also renders model interpretability more difficult. To overcome this problem a tree structure is proposed where the data is recursively partitioned into subsets, while each subset used to form models of lower levels of the tree. The concept is illustrated in Fig. 2.4, where the membership functions and the identified hinging hyperplane models are also shown.

During the identification the following phenomena can be taken into consideration (that can be considered as benefits too):

- By using hinging hyperplane function there is no need to find splitting variables at the nonterminal nodes, since this procedure is based on the hinge.
- Populated tree is always a binary tree either balanced, or non–balanced, depending on the algorithm (greedy or non–greedy). Based on binary tree, and the hinge splitting the $\mathbf{x}$ data pertains to left side of the hinge θ_1 always goes to the left child, and the right side behaves the same accordingly. For example given a simple symmetrical binary tree structure model, the first level

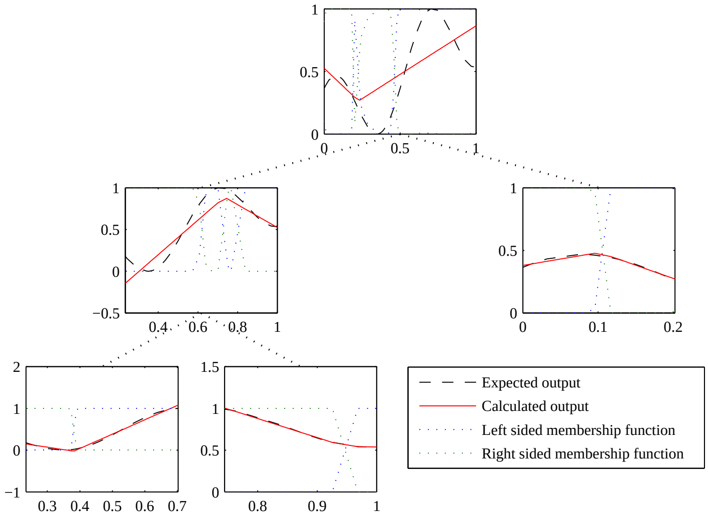

Figure 2.4: Hinging hyperplane based regression tree for basic data sample in case of greedy algorithm

contains one hinging hyperplane function, the second level contains 2 hinging hyperplane functions, the third level contains 4 hinges, and in general the kth level contains $2^{(k-1)}$ hinging hyperplane functions.

Concluding the above and obtaining the parameters θ during recursive identification the following cost function has to be minimized:

$$E(\{\theta_i\}, \pi) = \sum_{i=1}^{K} \pi_i E_{m_i}(\theta_i) \tag{2.21}$$

where K is the number of the hinge functions (nodes), and π is the binary ($\pi_i \in 0, 1$) terminal set, indicating that the given node is a final linear model ($\pi_i = 1$), and can be incorporated as a terminal node of the identified piecewise model.
Growing algorithm can be either balanced or greedy. In balanced case the identification algorithm builds the tree till the desired stopping criteria, while the greedy one will continue the tree building with choosing a node for splitting which performs worst during the building procedure. Hence, this operating regime needs further local models for better model performance. For a greedy algorithm the crucial item is the selection of the good stopping criteria. Any of the followings can be used to determine whether to continue the tree growing process or stop the procedure:

1. The loss function becomes zero. This corresponds to the situation where the size of the data set is less or equal to the dimension of the hinge. Since the

hinging hyperplanes are located by linear least–squares. From least–squares theory, when the number of data is equal to the number of parameters to be determined, the result would be exact, given the matrix is not singular.

2. $E = E_1 + E_2$, where $E_i = \sum_{k=1}^{N} \mu_{i,k} \big(y_k - f_i(\mathbf{x}_k; \theta_i)\big)^2$ represents the performances of the left and right hand side models of the hinge. During the growth of the binary tree, the loss function is always non-increasing, so E should be always smaller than the performance of the parent node. When no decrease is observed in loss function, when the tree growing should be stopped.

3. The tree building process reaches the pre-defined tree depth.

4. All of the identified terminal nodes performance meets an accuracy level (ε - error rate). In this case it is not necessary to specify the depth of the tree, but it can cause overfitting of the model.

The algorithm results are represented in Fig. 2.4 where $L = 3$, $K = 5$, and $\pi = [0, 0, 1, 1, 1]$. On Fig. 2.5 a 3–dimensional example is shown. The function

$$y = \frac{\sin\left(\sqrt{x_1^2 + x_2^2 + \epsilon}\right)}{\sqrt{x_1^2 + x_2^2 + \epsilon}} \tag{2.22}$$

has been approximated by hinging hyperplane based tree. On Fig. 2.5 it is shown how the approximation becomes much more smoother with applying 1,2, and 4 level and greedy building method. Not just the generation of the binary tree structured model is important, but to construct a greedy algorithm and to measure the identified model, node performance must be determined during the identification procedure, which can be defined in different ways:

- *Modeling performance of the nodes*

  The well–known regression performance estimators can be used for node performance measurement, in this work root mean squared prediction error (RMSE) was used.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2} \tag{2.23}$$

- *Condition of the nodes*

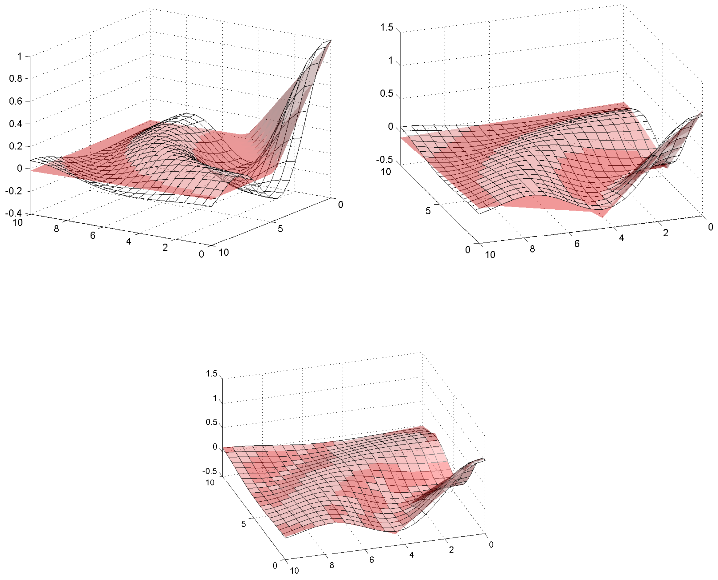

Figure 2.5: Modeling a 3D function with hinging hyperplane hyperplane based tree

> It is described that with two prototype clusters ($c = 2$) and a-priori knowledge (*constraints*) FCRM method is able to identify hinging hyperplanes, hence $\mu_{i,k}$ membership degree has information at a node about how many data points are going to the slitted prototypes. Comparing this data with the information about the hinge -based node splitting rule (how many datasamples are described by the θ_1, θ_2 parameter vectors) we can assign a certain condition (ϱ) to a node:
>
> $$\varrho_n = 1 - \frac{\|m_1 - \sum_{k=1}^{N_n} \mu_{1,k}\|}{\sum_{i=1}^{2} \sum_{k=1}^{N_n} \mu_{i,k}} \tag{2.24}$$
>
> where m_1 is the cardinality based on θ^+, while N_n represents the number of samples at node π_i.

We can consider ϱ as a measurement of the FCRM hinging hyperplane identification perfection. The closer ϱ is to 1 the better the identification. The hinge does not ”override” the $\mu_{i,k}$ membership degrees. This measure is very similar to the one, that was introduced in [50] and was used for identifying parameter similarity. In Figs. 2.6 and 2.7 RMSE and ϱ results of identifying Eq. 2.22 node-by-node(axis x) with the depth of 4 levels can be seen. On Fig. 2.6 non-greedy case can be examined while Fig. 2.7 shows performance of the greedy algorithm. It is visible that for tree building purposes cardinality based splitting is a very good approach.

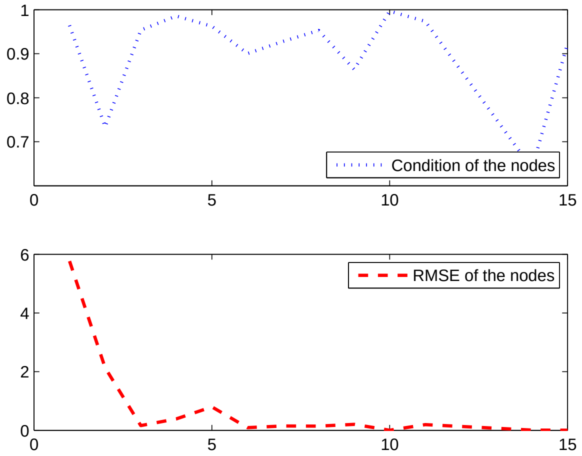

Figure 2.6: Node by node ϱ and RMSE results for non–greedy tree building

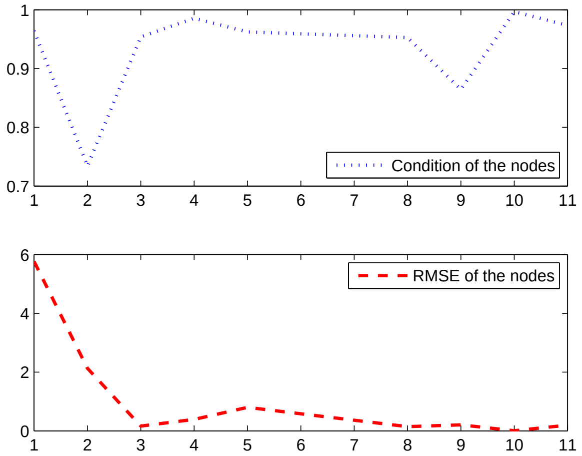

Figure 2.7: Node by node ϱ and RMSE results for greedy tree building

2.3 Application examples

Accuracy and transparency of the proposed algorithm are shown based on multiple datasets, two real life and two synthetic ones followed by examples in the area of dynamic system identification.

2.3.1 Benchmark data

All datasets have been used before, most of them are originated from well-known data repositories. Performance of the models is measured by the root mean squared prediction error (RMSE - see Eq. 2.23)

Table 2.1: Comparison of RMSE results of different algorithms. (Numbers in brackets are the number of models)

Data	Sample	HH	CART	FMID	FRT
Fried	Train	0.87	0.84	2.41(4)	0.69
	Test	0.92(8)	2.12(495.6)	2.41(12)	0.7 (15)
3Dsin	Train	.17	0.09	0.50(4)	0.18
	Test	0.18(11)	0.17(323.1)	0.31(12)	0.18(12)
Abalone	Train	2.62	1.19	2.20(4)	2.18
	Test	2,88 (8)	2.87(664.8)	2.19(12)	2.19(4)
Kinman	Train	0.15	0.09	0.20 (4)	0.15
	Test	0.16 (6)	0.23(453.9)	0.20(12)	0.15(20)

Real life datasets:

- *Abalone* Dataset from UCI machine learning repository[1] used to predict the age of abalone from physical measurements. Contains 4177 cases with 8 attributes (1 nominal and 7 continuous).
- *Kin8nm* Data containing information on the forward kinematics of an 8 link robot arm from the DVELVE repository. Contains 8192 cases with 8 continuous attributes.

Synthetic datasets:

- *Fried* Artificial dataset used by Friedman [51] containing 10 continuous attributes with independent values uniformly distributed in the interval [0,1]. The value of the output variable is obtained with the equation:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1) \tag{2.25}$$

- *3DSin* Artificial dataset containing 2 continuous predictor attributes uniformly distributed in interval $[-3, 3]$, with the output defined as

$$y = 3\sin(x_1)\sin(x_2) \tag{2.26}$$

3000 data points were generated using these equations.

For the robust testing of the performance of the model building algorithm, 10 fold cross validation method is utilized with data normalized to zero mean and unit

[1] FTP address: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

variance. Table 2.1 shows the performance on these datasets, compared to the results of other algorithms, can be found in [38]. Moreover Table 2.2 contains the min, mean, max error rates of the 10 fold cross validation results for the hinging hyperplane algorithm with the calculated standard deviation values as well. The comparative algorithms are fuzzy based (*FRT–Fuzzy Regression Tree, FMID–Fuzzy Model Identification*) and classical regression tree based also (*CART–Classification and Regression Tree*). It can be concluded that the performance of the introduced algorithm is in line with the other methods, also with moderate number of terminal nodes in the identified model tree. Results are consistent, even the worst performance of the 10–fold cross validation is in line.

Table 2.2: 10–fold cross validation report for hinging hyperplanes based tree

Data	**Sample**	**MIN**	**MEAN**	**MAX**	**Standard dev.**
Fried	Train	0.5822	0.8677	1.2107	0.227
	Test	0.6226	0.9208	1.2673	0.2337
3Dsin	Train	0.0906	0.1741	0.3162	0,0714
	Test	0.0838	0.178	0.342	0.0801
Abalone	Train	2.3496	2.6241	2.9256	0.1532
	Test	2.3242	2.8803	3.451	0.3445
Kinman	Train	0.1433	0.1515	0.1595	0.0054
	Test	0.1464	0.1579	0.1729	0.0092

2.3.2 Dynamic systems

The following subsection shows results on the identification first order non–linear dynamic system and describes performance of the proposed technique in model predictive control.

Identification of the Box-Jenkins gas furnace

The well–known Box–Jenkins furnace data benchmark is used to illustrate the proposed modeling approach and to compare its effectiveness with other methods. The data set consists of 296 pairs of input-output observations taken from a laboratory furnace with a sampling time of 9 seconds. The process input is the methane flow rate and the output is the percentage of CO_2 in the off gas. A number of researchers concluded that a proper structure of a dynamic model for this system is

$$y(k+1) = f(y(k), u(k-3)) \tag{2.27}$$

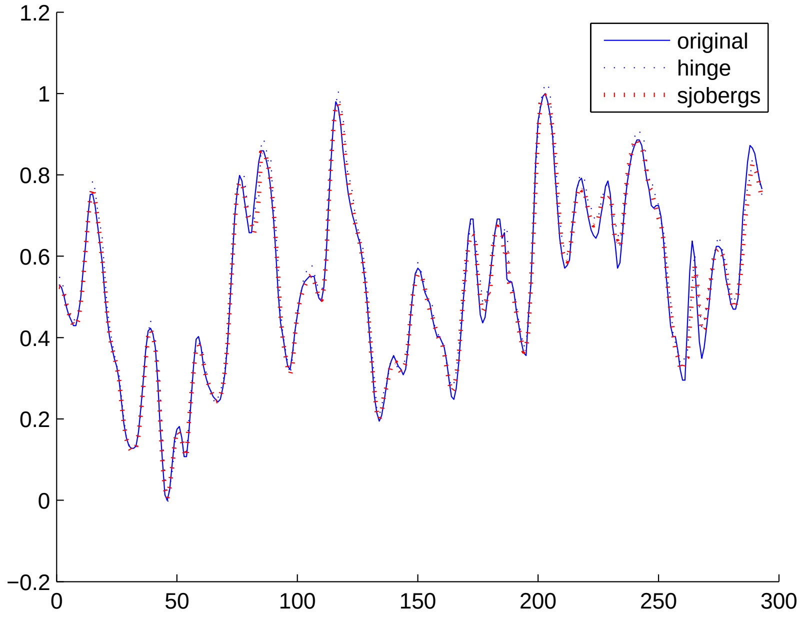

Figure 2.8: Identification of the Box–Jenkins gas furnace model with hinging hyperplanes

The approximation power of the model can be seen in Fig. 2.8 and table 2.3. Comparing results with other techniques referred in [52] it can be concluded that modeling performance is in line with other techniques with moderate number of identified hinging hyperplanes.

Table 2.3: RMSE results of the generated models

Method	**Training**	**Testing**	**Free Run**	**HH ♯**
Proposed technique	0.0266	0.0311	0.0374	4
Sjoberg model	0.0336	0.0342	0.0351	4

So far, *a general nonlinear modeling technique* was presented and a new identification approach was given for hinging hyperplane based nonlinear models: $\widehat{y} = f(\mathbf{x}(k), \theta)$ where $f(.)$ represents the hinging hyperplane based tree structured model and $\mathbf{x}(k)$ represents the input vector of the model. To identify a discrete-time *input-output model* for a dynamical system, the dynamic model structure has to be chosen or determined beforehand. A possible often applied structure is nonlinear autoregressive model with exogenous input (NARX) where the input vector of the model $\mathbf{x}(k)$ contains the delayed inputs and outputs of the system to be modeled [32]. In several practical cases a simpler and more specific model structure can be used to approximate the behavior of the system, which fits better the structure of the system. Therefore, it can be an advantage for the identification approach (models with simpler structure can be identified easier), and this model can be more accurate. One such special case of the NARX model is the Hammerstein model, where the same static nonlinearity f is defined for all of the delayed control inputs (for the sake of simplicity, SISO models are considered):

$$\widehat{y} = \sum_{i=1}^{n_a} a_i y(k-i) + \sum_{j=1}^{n_b} b_j f(u(k-j)) \tag{2.28}$$

where $y()$ and $u()$ are the output and input of the system, respectively, and n_a and n_b are the output and input orders of the model. The parameters of the blocks of the Hammerstein model (static nonlinearity and linear dynamics) can be identified by the proposed method simultaneously if the same linear dynamic behavior can be guaranteed by all of the local hinging hyperplane based models. It can be done in an elegant way utilizing the flexibility of the proposed identification approach: global constrains can be formulated for the a_i and b_j parameters of the local models (for a detailed discussion what constrains have to be formulated, see [32]). In the following, the hinging hyperplane modeling technique is applied on a Hammerstein type system. It will be shown why it is an effective tool for the above–mentioned prupose.

Model predictive control

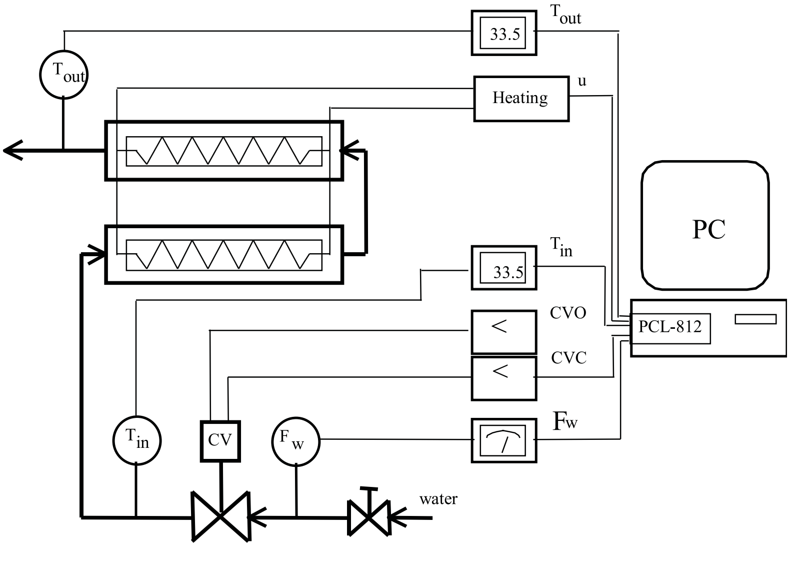

Figure 2.9: The water heater

Modeling of a simulated water heater (Fig. 2.9) is used to illustrate the advantages of the proposed hinging hyperplanes based models. The water flows through a pair of metal pipes containing a cartridge heater. The outlet temperature, T_{out}, of the water can be varied by adjusting the heating signal, u, of the cartridge heater (see [32] or Appendix E for details). The performance of the cartridge heater is given by:

$$Q(u) = Q_M \left[u - \frac{\sin(2\pi u)}{2\pi} \right] \tag{2.29}$$

where Q_M is the maximal power and u is the heating signal (voltage). As the equation above shows the heating performance is a static nonlinear function of the heating signal. Hence, the Hammerstein model is a good match to this process. The aim is to construct a dynamic prediction model from data for the output temperature (the dependent variable, $y = T_{out}$) as a function of the control input: the heating signal. The parameters of the Hammerstein model were chosen as $n_a = n_b = 2$. The performance of this modeling technique will be compared to linear and feedforward neural network models. The modeling performances can be seen in Table 2.4. Modeling errors were also calculated based on RMSE (see Eq. (2.23)). In this example a hinging hyperplane function based tree with 4 leaves were generated. For robust testing of the model building algorithm performance, 10–fold cross validation method is used. For comparison, a feedforward neural net and linear model was also trained and tested using the same data. The neural net contains one hidden layer with 4 neurons using $\tanh$ basis functions. As can be seen from the results, the training and test error are comparable with the errors of the proposed method. A very rigorous test of NARX models is free run simulation because the errors can be cumulated. It can be also seen in Fig. 2.10 that the identified models (the pro-

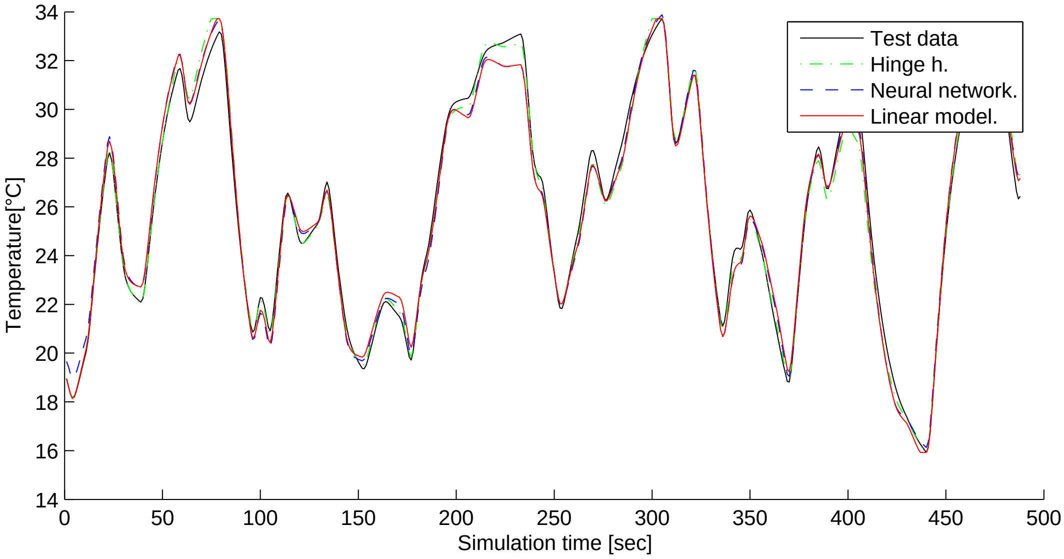

Figure 2.10: Free run simulation of the water heater (proposed hinging hyperplane model, neural network, linear model)

posed ones, linear models and the neural nets) perform very good also in free run simulation (the system output and the simulated ones can hardly be distinguished). Although the neural net seems to be more robust in this example, the proposed hinging hyperplane model is much more interpretable than the neural net [1]. This confirms that both the proposed clustering based constrained optimization strategy and the hierarchial model structure has advantages over the classical gradient-based optimization of global hinging hyperplane models.

Table 2.4: RMSE results of the generated models

Method	**Training**	**Testing**	**Free Run**
Linear model	0.0393	0.0449	0.387
Neural network	0.0338	0.0403	0.356
Proposed method	0.0367	0.0417	0.359

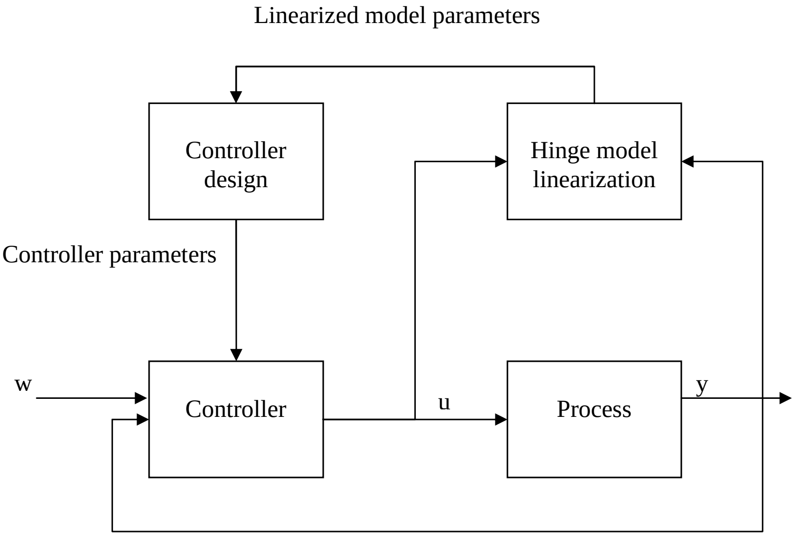

Figure 2.11: Structure of the MPC controller

In the following this model will be applied for model predictive control. Details of model-based control of fuzzy and operating regime models can be found in [42] and [43]. Among the wide range of possible solutions a model predictive controller (MPC) was designed. Fig. 2.11 shows the structure of an MPC controller.

Real time control needs low computational complexity. Hence a time varying linear MPC is designed based on time varying parameters of a linear model extracted in every time instant from the regression tree. This scheme is widely studied and similarities to the nonlinear optimization based control solutions including convergence were also shown. In [32] it was shown that there are two options to obtain a linear model from a nonlinear operating regime based (fuzzy) model. The Taylor-expansion based linearization assumes the global interpretation of the model, while the linear parameter varying (LPV) linear model extraction approach considers the model as an interpolating system between local linear time-invariant (LTI) models were the dynamic effect of the interpolation is negligible. Fortunately, thanks to hinging hyperplane models and tree structured representation the proposed model perfectly supports this interpretation, hinging hyperplanes define local linear models and their operating regimes. Since these local models do not overlap, the negative effect of the interpolating functions do not have to taken into account.

The classical model predictive controller computes an optimal control sequence by minimizing the following cost quadratic cost function:

$$J\left(H_p, H_c, \lambda\right) = \sum_{j=1}^{H_p} \left(w\left(k+j\right) - \hat{y}\left(k+j\right)\right)^2 + \lambda \sum_{j=1}^{H_c} \Delta u^2\left(k+j-1\right) \tag{2.30}$$

where, $\hat{y}\left(k+j\right)$ denotes the predicted process output, H_p is the maximum costing or prediction horizon, H_c is the control horizon, and λ is a weighting coefficient. According to the receding horizon principle only the first element of the optimized control sequence is applied $u(k)$, and this optimization is performed in every time instant. This scheme allows real time control, feedback of model errors, handles unmeasured disturbances, and supports the previously mentioned iterative liberalization scheme. Details about the convergence and possible extensions of this control scheme can be found in [33]. The key equation of MPC is the prediction of the model:

$$\hat{\mathbf{y}} = \mathbf{S}\Delta\bar{\mathbf{u}} + \mathbf{p} \tag{2.31}$$

where the model prediction equation is given in its vector–based form as $\Delta\bar{\mathbf{u}} = [\Delta u(k), \ldots, \Delta u(k+H_c)]$, and $\mathbf{p} = [p_1, p_2, \ldots, p_{Hp}]$ and $\hat{\mathbf{y}} = [\hat{y}(k+1), \ldots, \hat{y}(k+H_p)]$ and the $\mathbf{S}$ containing the parameters of a step-response model is an $(H_p) \times H_c$ matrix with zero entries $s_{i,j}$ for $j-i>1$:

$$\mathbf{S} = \begin{bmatrix} s_1 & 0 & 0 & 0 \\ s_2 & s_1 & 0 & 0 \\ \vdots & & \ddots & \\ s_{H_p} & s_{H_p-1} & \cdots & s_{H_p-H_c} \end{bmatrix}. \tag{2.32}$$

When constraints are considered, the minimum of the cost function can be found by quadratic optimization with linear constraints:

$$\begin{gathered}\min_{\bar{\mathbf{u}}}\left\{(\mathbf{S}\Delta\bar{\mathbf{u}}+\mathbf{p}-w)^T(\mathbf{S}\Delta\bar{\mathbf{u}}+\mathbf{p}-w)+\lambda\Delta\bar{\mathbf{u}}^T\Delta\bar{\mathbf{u}}\right\}\\ \min_{\bar{\mathbf{u}}}\left\{\tfrac{1}{2}\boldsymbol{\Delta}\bar{\mathbf{u}}^T\mathbf{H}\Delta\bar{\mathbf{u}}+\mathbf{d}\Delta\bar{\mathbf{u}}\right\}\end{gathered} \tag{2.33}$$

with $\mathbf{H}=2\left(\mathbf{S}^T\mathbf{S}+\lambda\mathbf{I}\right)$, $\mathbf{d}=-2\left(\mathbf{S}^T\left(\mathbf{w}-\mathbf{p}\right)\right)$, where $\mathbf{I}$ is an $(H_c\times H_c)$ unity matrix.

The constraints defined on u and Δu can be formulated with the following inequality:

$$\begin{pmatrix}\mathbf{I}_{\Delta\bar{\mathbf{u}}}\\ -\mathbf{I}_{\Delta\bar{\mathbf{u}}}\\ \mathbf{I}_{H_c}\\ -\mathbf{I}_{H_c}\end{pmatrix}\Delta\bar{\mathbf{u}}\leq\begin{pmatrix}\mathbf{u}_{\max}-\mathbf{I}_{\bar{\mathbf{u}}}u(k-1)\\ -u_{\min}+\mathbf{I}_{\bar{\mathbf{u}}}u(k-1)\\ \boldsymbol{\Delta}u_{\max}\\ -\Delta u_{\min}\end{pmatrix} \tag{2.34}$$

where $\mathbf{I}_{H_c}$ and $\mathbf{I}_{\bar{\mathbf{u}}}$ is an $H_c\times H_c$ unity matrix, $\mathbf{I}_{\boldsymbol{\Delta}\bar{\mathbf{u}}}$ is an $H_c\times H_c$ lower triangular matrix with all elements equal to one, and $\boldsymbol{\Delta}u_{\min}$,$\Delta u_{\max}$,$u_{\min}$,$u_{\max}$ are H_c-vectors, with the constraints $\Delta u_{\min}$, $\Delta u_{\max}$, $u_{\min}$, $u_{\max}$ respectively.

To handle modeling error the MPC is applied in the well–known internal model control (IMC) scheme where the setpoint of the controller is shifted by the filtered modeling error. For this purpose a first-order linear filter is used:

$$e_{mf}(k)=\alpha e_m(k)+(1-\alpha)\,e_{mf}(k-1)\,, \tag{2.35}$$

where $0\leq\alpha<1$ is determined such that a compromise between performance and robustness is achieved. Effective suppressing of the steady-state modeling error can be achieved by a proper tuning of this filter. The best parameters are found for the controller: $H_p=9$, $H_c=2$, $\lambda=20$ and $\alpha=0.95$. Simulation results for Hinging hyperplane based model, the affine neural network model and the linear model are shown in figures 2.12, 2.13, 2.14.

At the operation region edges, the MPC based on the linear model resulted undesirable overshoots and undershoots. This is a direct consequence of a bad estimation of the nonlinear gain of the system in these regions. This over-estimation of the system gain by the linear model is also seen in the sluggish control action. In contrast, the MPC based on the nonlinear models shows a superior performance over the whole operating region. Among these, the MPC based on the hinging hyperplane model results in the smallest overshoot with the fastest change in the control signal.

Notice also that the oscillatory behavior of the neural network model based MPC

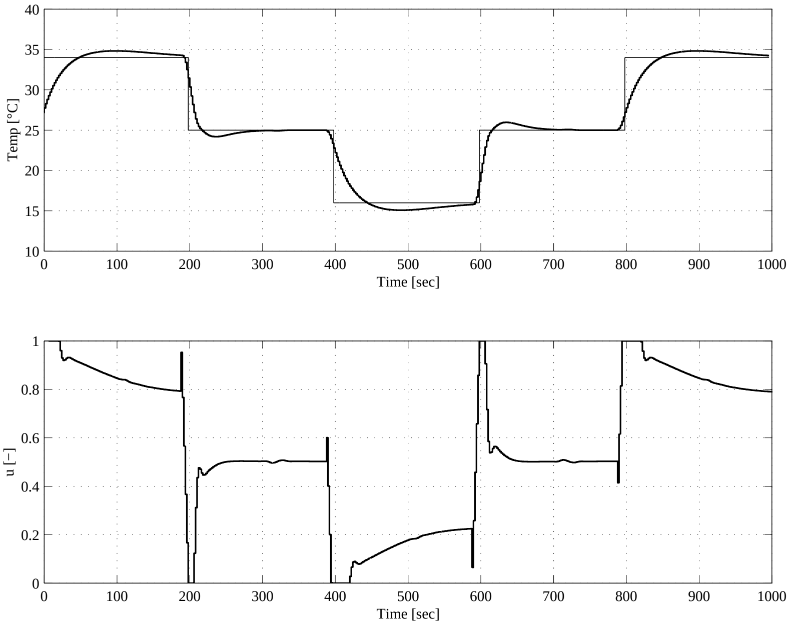

Figure 2.12: Performance of the MPC based on linear model

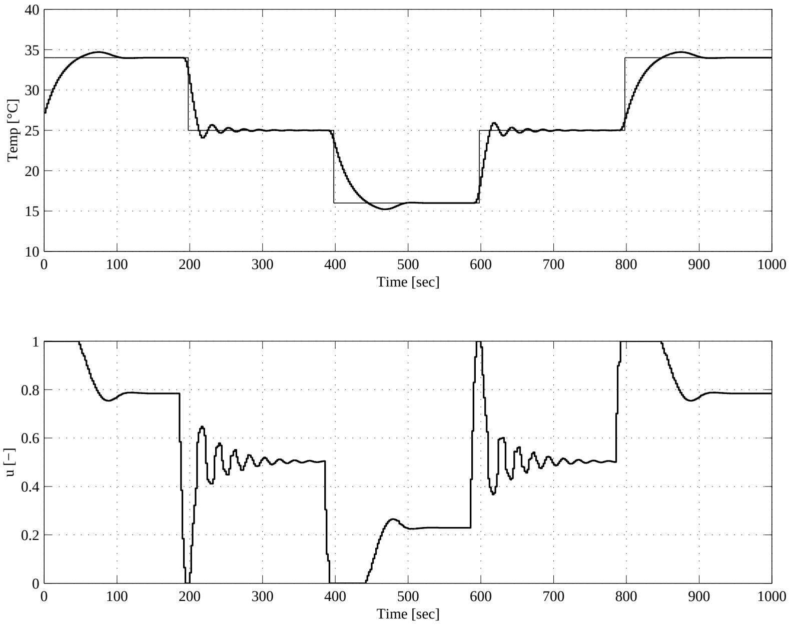

Figure 2.13: Performance of the MPC based on Neural Network model

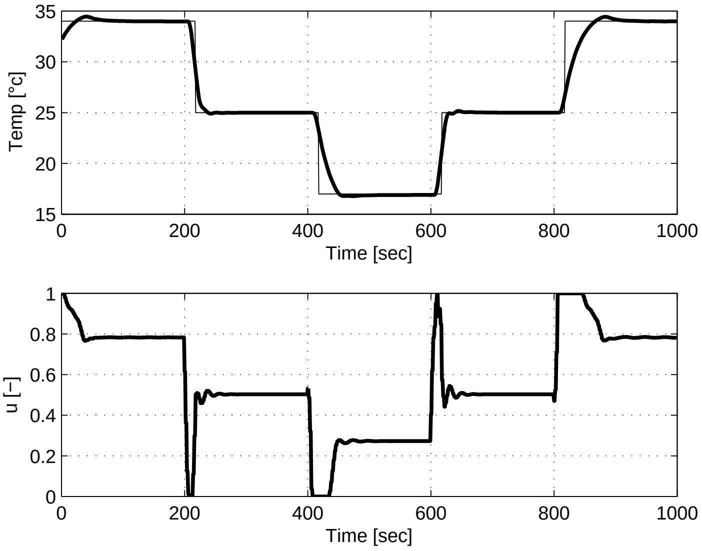

Figure 2.14: Performance of the MPC based on hinging hyperplane

is due to the bad prediction of the steady-state gain of the system around the middle region. However, as can be seen from Table 2.5, both nonlinear models achieved approximately the same summed squared tracking error (SSE), although a smaller control effort (CE) was needed for the hinging hyperplane based MPC.

The applied model in GPC	SSE	CE
Linear model	1085	1.61
Neural Network model	956	1.39
Hinging hyperplane model	966	0.58

Table 2.5: Simulation results (SSE - sum squared tracking error, CE - sum square of the control actions)

2.4 Conclusions

A novel framework for hinging hyperplane based data driven modeling has been developed. Fuzzy c-regression clustering can be used to identify the parameters of two hyperplanes. Hierarchical regression tree is obtained by the recursive clustering of the data. The complexity of the model is controlled by the proposed model model performance measure. The resulted piecewise linear model can effectively used to represent nonlinear dynamical systems. The resulted linear parameter varying (LPV) model can be easily utilized in model based control.

To illustrate the advantages of the proposed approach, benchmark datasets were modeled and simulation example is presented for the identification and model predictive control of a laboratory water-heater.

The results show that with the use of the proposed modeling framework accurate and transparent nonlinear models can be identified since the complexity and the accuracy of the model can be easily controlled. The local linear models can be easily interpreted and utilized to represent operating regimes of nonlinear dynamical systems. Based on this interpretation, effective model based control applications can be designed.

Chapter 3

Neural Networks

Neural network itself is a black-box model, so it doesn't reveal any information about the identified system. It is challenging task to open up this box to support model building procedures. However, based on the extracted information model reduction and visualization could be done on the base model. The key idea is that the neural networks can be transformed into a fuzzy rule base where the rules can be analyzed, visualized, interpreted and even reduced.

Section 3.1 shows how NNs work. This description is mainly based on [31], for a detailed discussion see [1]. Having the main concepts set for NNs Section 3.2 gives a brief introduction and overview about the applied NN transformation method which means the basis for model reduction and visualization. Section 3.3 contains a combined approach used in this thesis to get reduced rule based model from NN. Section 3.4 overview some NN visualization method, and propose a new technique to measure the similarity of neurons which gives the basis of the visualization approach. In Section 3.5 some illustrative examples are given, and Section 3.6 concludes the chapter.

3.1 Structure of Neural Networks

The systematic study of continuous functions started in the nineteenth century. The complexity of this function class was demonstrated by Weierstrass famous example; *the everywhere continuous but nowhere differentiable real function*[53].

Theoretical foundations of the function approximation was also done by Weierstrass: he showed in that a*ny continuous function on a closed real interval could uniformly approximated by polynomials*[54]. Another question was the theory of multivariate continuous function approximation. Around 1900 Hilbert suspected the existence of continuous multivariate functions, which cannot be approximated by arbitrarily chosen continuous univariate functions[55]. Counter-examples for Hilbert's conjecture arose only in 1957. Arnold ternary continuous functions [56], later constructive example was given by Kolmogorov for the approximation of multivariate continuous functions by continuous univariate functions[57].

Later the Kolmogorov method was simplified in the early sixties of the last century (eg, Sprecher [58] and Lorentz [59]), it has become to the theoretical basis of universal approximators of continuous functions. It was revealed in the eighties and nineteens years of the last century that feedforward multilayer neural networks (eg, [60, 61, 62, 63] and fuzzy systems constructed by Zadeh [64] are universal approximators [65, 66, 67], although the limitations of the approximation capabilities of systems constructed from finite components become clear[68, 69, 70].

Around the middle fifties Marshall Harve Stone further generalized Weierstrass statements [71, 72], so the Stone-Weierstrass theorem is mentioned in respect to neural networks[73, 74].

3.1.1 McCulloch-Pitts neuron

McCulloch and Pitts in 1943 developed a simple mathematical model for a neuron (3.1).

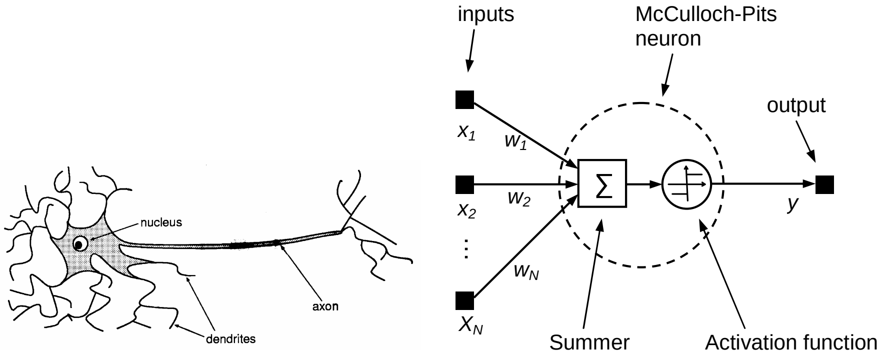

Figure 3.1: A biological neuron and its model (McCulloch-Pitts neuron)

The McCulloch-Pitts neuron has multiple inputs and a single output. Each of the inputs has an associated weight. Weighted sum of the inputs is passed through a nonlinearity to the output of the neuron as follows:

$$y = f\left(\sum_{i=1}^{N} \mathbf{w}_i \mathbf{x}_i\right), \begin{cases} 0 & = & z < 0 \\ 1 & = & z \geq 0 \end{cases} \tag{3.1}$$

where x_i are the inputs and y is the output of the neuron, $f(z)$ is the non-linear activation function in the form of the step function given above, and w_i are the strengths of the connections or weights. Multi-layer neural networks is a network of neurons bunched together in a multiple layers network. A feedforward neural network has one input layer, one output layer and a number of hidden layers between them. Normally we use neural networks with one hidden layer. This model is very general. It has been shown that with one hidden layer a network can describe any continuous function (if there are enough hidden units), and that with two hidden layers it can describe any function at all. Detailed description of neural network structures, utilization and activation function types can be found [1, 17].

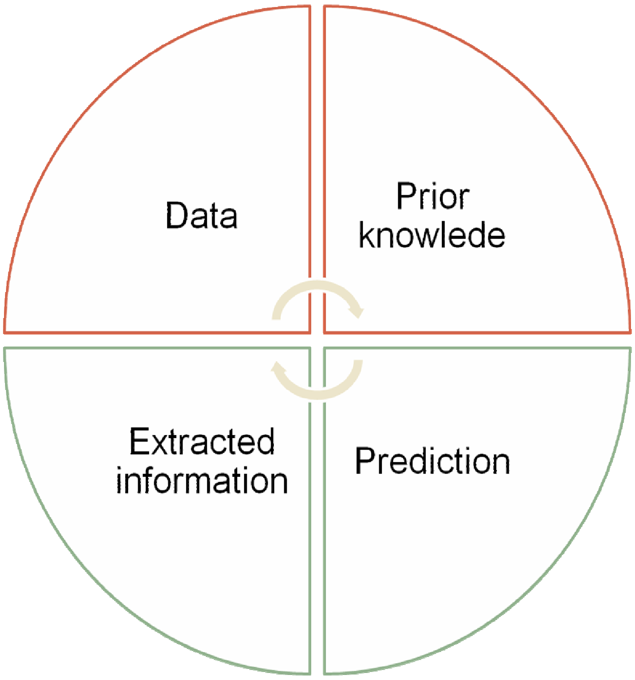

Figure 3.2: Modeling framework

Based on these model types motivation of our work is to prepare a tool where data, prior knowledge, prediction and extracted information (see Fig. 3.2) forms an integrated framework to help model building procedures with interpretability, visualization and reduction of multilayer perceptron (MLP – usually one hidden layer is applied) type neural networks with logistic hidden activation function.

3.2 NN Transformation into Rule Based Model

A possible strategy for 'opening' a NN is to convert it into a rule based model. These 'linguistically sound' rules are often fuzzy if-then rules, and are close to human thinking: IF *a set of conditions is satisfied*, THEN *a set of consequences is inferred.* Fuzzy logic provides a tool to process uncertainty, hence fuzzy rules represents knowledge using linguistic labels instead of numeric values, thus, they are more understandable for humans and may be easily interpreted [31]. If NNs can be transformed into rules, then it makes possible to overlook and validate the trained NN, and build in a priori knowledge to the network. The crucial question is what the connection is between the several types of neural networks and fuzzy rule based systems.

Under some conditions, the equivalence of normalized radial basis function networks (RBF) and Takagi-Sugeno fuzzy models can be obtained [1]. However, in this thesis, multilayer perceptron (MLP) type neural networks with logistic hidden activation function are used (in the following the notation NN will be used for MLP type networks). An approach for NNs with $\tanh$ activation function is presented in [75] for function approximation purposes, but it should be noted that it is an approximation: the rule based model is not identical to the original trained NN, therefore information transfer in the 'opposite' direction, i.e. from the rule base to the NN can be problematic. An interesting result was given in [31] where the equality of NNs with logistic activation function and a certain type of fuzzy rule based model called fuzzy additive system (FAS) was proven. For that purpose, a new fuzzy logic operator had to be introduced. Because of the equality (which is stronger than equivalence), if a method can be applied on a FAS for a certain purpose (e.g. rule base reduction), then it is also applicable to the NN as well and vice versa.

3.2.1 Rule–based interpretation of neural networks

In the following, this equality relation is discussed based on [31]. FAS employs rules in the following form:

$$\mathbf{R}_{jk}: \textbf{ If } x_1 \; is \; A_{jk}^1 \; and \ldots and \; x_n \; is \; A_{jk}^n \textbf{ then } y_k \; is \; \delta_{jk}\left(x_1, \ldots, x_n\right) \tag{3.2}$$

where $\delta_{jk}\left(x_1, \ldots, x_n\right)$ is a linear function of the inputs. In FAS's, the inference engine works as follows: for each rule, the fuzzified inputs are matched against the corresponding antecedents in the premises giving the rule's firing strength. It is obtained as the t-norm (usually the minimum operator) of the membership degrees on the rule if-part. The overall value for output y_k is calculated as the weighted sum of relevant rule outputs. Let us suppose multi-intput single-output fuzzy rules, having l_k of them for kth output. Then y_k is computed as

$$y_k = \sum_{j=1}^{l_k} \beta_{jk} \delta_{jk} \left(x_1, \ldots, x_n\right) \tag{3.3}$$

where β_{jk} is the firing strength of j th rule for k th output.

To decompose the multivariate logistic function to form the rule antecedents in the form of eq. 3.2 with *univariate* membership functions, a special logic operator has to be used instead of and : interactive-or or i-or:

$$a * b = \frac{(ab)}{(1-a)(1-b)+ab} \tag{3.4}$$

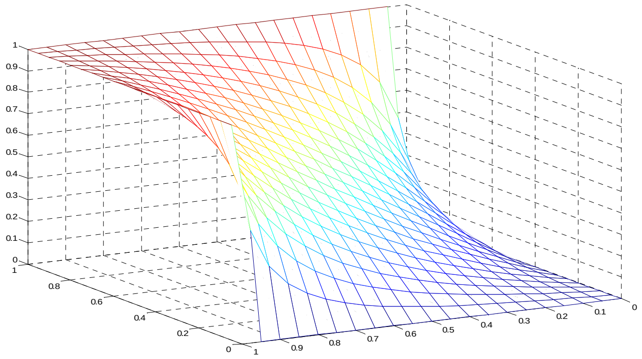

Figure 3.3: Interactive or operator

To get a clearer idea of i-or behavior, see Fig. 3.3, which represents the surface defined by the i-or operator. Using this $*$ operator, the interpretation of NNs whose hidden neurons have biases as follows. It can be checked that

$$f_A\left(\sum_{i=1}^{n} x_i w_{ij} + \tau_j\right) = f_A\left(x_1 w_{1j} + \tau'_j\right) * \ldots * f_A\left(x_n w_{nj} + \tau'_j\right) \tag{3.5}$$

where $\tau'_j = \tau_j / n$ and the first term corresponds to the fuzzy proposition ”$\sum_{i=1}^{n} x_i w_{ij} + \tau_j$ is A”. Likewise, $f_A\left(x_i w_{ij} + \tau'_j\right)$ corresponds to proposition ”$x_i w_{ij} + \tau'_j$ is A” or in a similar form ”$x_i w_{ij}$ is $A - \tau'_j$”. Hence, the bias term means a sheer translation. The A^i_{jk} fuzzy sets have to be redefined to account for both the weight w_{ij} and the bias τ'_j. Their membership function is defined by (see Fig. 3.4 for better explanation):

$$\mu_{A^i_{jk}}(x) = \mu_A\left[\left(x + \tau_j'\right) * w_{ij}\right] \tag{3.6}$$

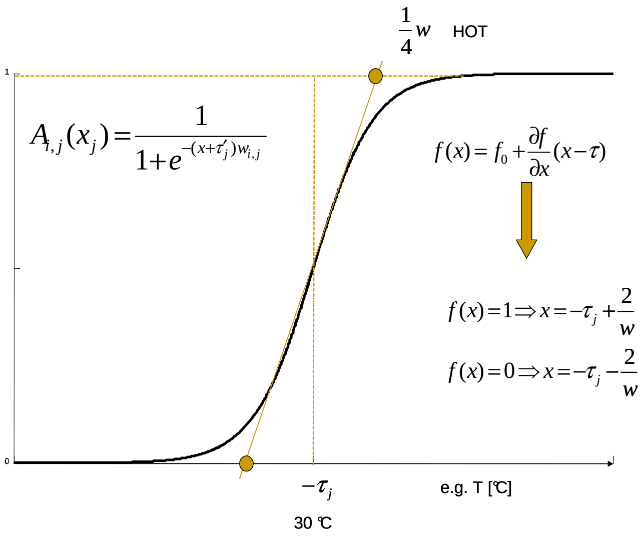
$\frac{1}{4}w$
HOT
1
$A_{i,j}(x_j)=\frac{1}{1+e^{-(x+\tau_j')w_{i,j}}}$
$f(x)=f_0+\frac{\partial f}{\partial x}(x-\tau)$
$f(x)=1 \Rightarrow x=-\tau_j+\frac{2}{w}$
$f(x)=0 \Rightarrow x=-\tau_j-\frac{2}{w}$
0
$-\tau_j$
e.g. T [℃]
30 ℃

Figure 3.4: Interpretation of the activation function

Based on that, the fuzzy rules extracted from the trained NN are:

$$\mathbf{R}_{jk} : \ \mathbf{If}\ x_1\ is\ A^1_{jk}\ * \ldots and *\ x_n\ is\ A^n_{jk}\ \mathbf{then}\ y_k\ =\ \delta_{jk} \tag{3.7}$$

An interesting and useful application possibility is to initialize the NN on the basis of a priori knowledge. Initialization is a crucial question by NNs because there are often a huge number of parameters and the cost function has numerous local minima. The most often applied local (gradient based) search techniques may trap in a local minimum. To avoid that problem, a possible approach is multi-start method, i.e. to train the NN from several different (random) initial points. Other solution can be based on evolutionary algorithms, see [76]. The flexibility of evolutionary algorithms makes possible the direct rule extraction from trained NNs (however, only crisp rules and by classification problems) as [77] shows. However, all of these latter methods have high computational demand. The initialization using prior knowledge based if-then rules has other advantage as well: it combines the user's experience with the learning capability of NN.

3.3 Model Complexity Reduction

In this section we focus on the combination of existing model reduction techniques with the previously presented rule based model extraction method. An interesting solution to NN reduction is the following: the complexity of the model is penalized, and it is built-in to the training procedure. The method proposed in [78] uses a cost function that consists of two terms: one for the NN accuracy (like mean square error) and one related to the NN complexity (numbers and magnitude of parameters). However, determination of their weights or relative importance is problematic. A weighting factor is introduced and several NNs should be trained with different weighting parameters. To compare the trained NNs and choose the best one, [78] applied the predicted square error measure.

In case of MLP networks and FAS systems classical OLS (see appendix C for further details) can be applied on FAS systems to rank the rules since the parameters of the trained NN are fixed. However, OLS is formulated as a MISO technique. If the NN has more than one output, then the outputs can be evaluated individually one by one. In this case (using the notation of OLS (eq. C.1- eq. C.3), y is the kth network output, the regressors z_i are the outputs of the hidden neurons, and the parameters θ_j corresponds to the weights from the jth hidden neuron to the kth output neuron β_{jk}. This approach was directly applied on NNs in [30], and it was shown that analog method can be applied to the subset selection of the original network inputs. In this case in eq. C.1- eq. C.3, y is the output of the kth hidden neuron, the regressors z_i are the inputs of the network, and the parameters θ_j corresponds to the weights from the jth input neuron to the kth hidden neuron w_{jk}. Other NN pruning can also be considered, e.g. optimal brain damage [26] or optimal brain surgeon [79], and it should be emphasized that these methods can directly be applied on FAS systems as well. The application examples in Section 3.5 show that it can be very effective if a model reduction technique and rule base extraction from NN are applied together, and validate the identified models by human experts.

Note that ordering the neurons by OLS estimated error reduction ratios reveals the unnecessary neurons (the importance of the extracted rules) in the hidden layer, because neurons with low error reduction ratio are insignificant for the appropriate model. As the equality of FAS and NN was proven in [31] and was discussed also in Section 3.2, the OLS ranking means a reduction based on the *consequent of the fuzzy rule*.

It should be noted that the applied i-or operator in the extracted fuzzy rules does not belong to the commonly applied fuzzy t-norms or t-conorms. However, it would be interesting to test the extracted fuzzy rules with common fuzzy logic operators, and maybe recompute the output weights (which can easily be done because the model is linear with respect to these parameters). Our presumption is that the crisper the activation functions are (f_A), the less the difference is between the modeling performances of the original and the modified FAS's that uses classical fuzzy logic operators. For that purpose, numerous tests have to be completed in the future. If this guess proves true, then the cost function for NN training can be modified to get 'crisper' activation functions.

3.4 NN Visualization Methods

In this section, a new technique for the visualization of neural networks is proposed. First, methods are discussed that can directly be applied on NNs. Second, a new approach is presented to detect the redundant neurons based on their similarity. This method exploits the equality of NNs and FAS's because it is based on the similarity of fuzzy membership functions.

The output of hidden neurons z_j can be seen as a h dimensional vector that represents the range the neurons work in. If a 'hidden variable' z_j is close to zero or one, the neuron is saturated. If a hidden neuron gives values near zero or one for almost all inputs, hence it does not fire or fires all the time, it is useless for the problem. The distribution of these h dimensional data can represent the NN behavior for a human expert. Unfortunately, in several cases there is a need for more than two or three hidden neurons. In these cases a projection or dimensionality reduction technique has to be used. Principal Component Analysis (PCA) is a linear technique; therefore the information loss may be more than the admissible level. Other (topology or distance preserving) projection techniques like Multidimensional Scaling, Sammon method, Isomap or Locally Linear Embedding can be used for that purpose. For more details see [32] and the references within.

However, there are some special visualization methods for NNs. Duch [27] proposed an approach for visualization of NNs applied on classification problems. His method can be applied for problems with K classes if the output is coded as a K length vector: $(1, 0, \ldots, 0)$ means the first class, $(0, 1, \ldots, 0)$ the second and so on. In this case case the classes are represented by the corners of the K dimensional unit hypercube. The approach proposed by Duch maps the NN *output* into two dimensions, basically 'flattens' the hypercube into two dimensions. This approach was thought over and applied on the *output of the hidden neurons* z_j in [28, 29]. This method was straightforward from the former one because the hidden variables (the activation functions) take values from $[0, 1]$, therefore the h dimensional vectors are located within the unit hypercube. This method can be used not only for classification but also for function approximation purposes as well. Based on this latter approach a picture of the behavior of the hidden units, their firing strength and activation or saturation level can be obtained. The main drawback is that the number of classes/hidden neurons is limited. To keep the figures simple and interpretable, only $3 \dots 6$ variables can be used.

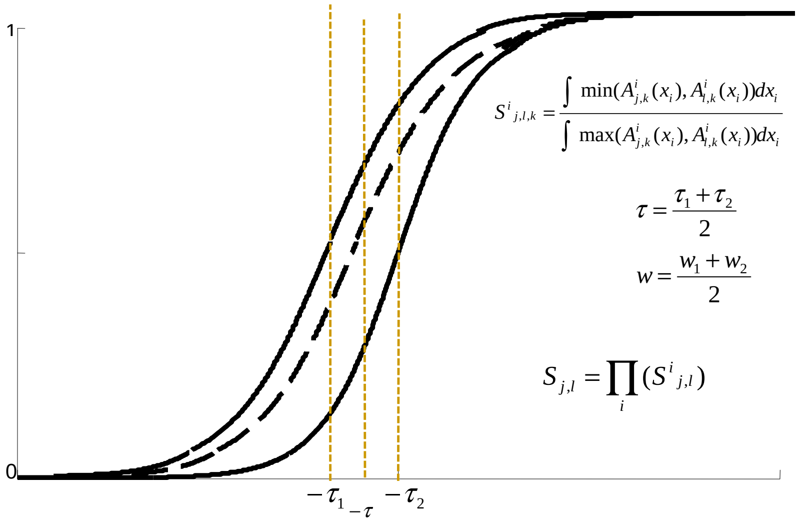

Figure 3.5: Similarity index

In the following, a different method is proposed to visualize the NN. The presented approach utilizes the *antecedent part* of the extracted fuzzy rules (since OLS based model reduction uses the consequent parts, see Section 3.3). To reduce the FAS rule base by analyzing the antecedent part of the rules is possible with measuring the similarity of the membership functions, and removing the too similar neurons. Utilizing the equality of FAS and NN, the following classical interclass separability measure (originally for fuzzy systems) could be used to compare the univariate functions decomposed from hidden neurons:

$$S^i_{j,l,k} = \frac{\int \min\left(A^i_{j,k}\left(x_i\right), A^i_{l,k}\left(x_i\right)\right) dx_i}{\int \max\left(A^i_{j,k}\left(x_i\right), A^i_{l,k}\left(x_i\right)\right) dx_i} \tag{3.8}$$

where $i = 1, \dots, n,\ j, l = 1, \dots, h.$. Eq. 3.8 can be used to measure the similarity of two *clauses* in the rule base, in other words the similarity of two hidden neurons for the same input variables (see Fig. 3.5). To compare the hidden neurons themselves with multivariate activation functions, the following measure seems to be straightforward:

$$S_{j,l,k} = \prod_{i} S^{i}_{j,l,k}, \; i = 1, \ldots, n, \; j, l = 1, \ldots, h. \tag{3.9}$$

With this measure, pairwise similarities of hidden neurons in the range of $[0, 1]$ can be obtained. To get pairwise distances if needed, the simple form of $1 - S_{j,l,k}$ can be used. Based on these distances which can be called relative data, the neurons themselves can be mapped onto two dimensions. In this thesis, the classical multidimensional scaling will be used. This well-known technique is not discussed here because it would exceed the size and scope of this thesis. The mapped two dimensional points refer how similar the neurons behave. As can be seen, the above mentioned approaches [28, 29] visualize the output of the hidden neurons, and draw conclusions from the location of these data. The proposed approach focuses to the behavior of hidden neurons as well, but utilizes the shape of the identified multi-dimensional activation functions. The previous approach can be used to determine how well the NN was trained, since the proposed one shows which neurons are similar and redundant within the trained network. In this formulation, this method can rather be used for complexity reduction purposes, and not to qualify the training procedure.

3.5 Application Examples

3.5.1 pH process

For applying the introduced visualization and reduction techniques we used a dataset of a pH process (see [32] or Appendix D), where the concentration of hydrogen ions in a continuous stirred tank reactor is modeled (CSTR). This well-known modeling problem presents difficulties due to the nonlinearity of the process dynamics. This process can be correctly modeled as a first-order input-output (NARX) system, where the actual output (the pH) $y(k+1)$, depends on pH of the reactor $y(k)$ and the NaOH feed $u(k)$ at the kth sample time (sample time is $t_s = 0.2min$):

$$y(k+1) = f\left(y\left(k\right), u\left(k\right)\right) \tag{3.10}$$

Parameters of the neural network were identified by the back-propagation algo-

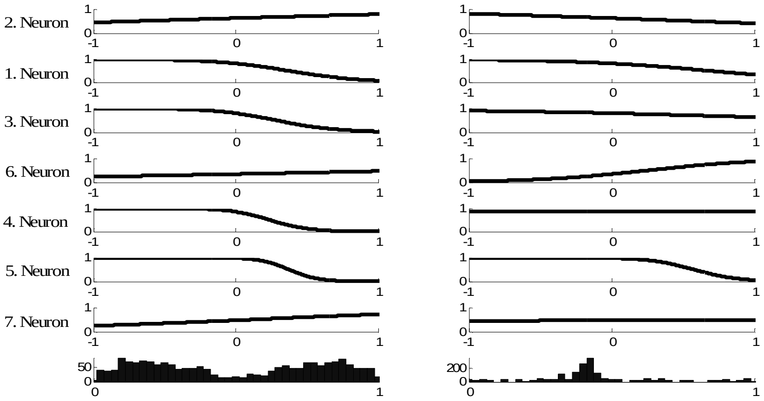

Figure 3.6: Decomposed univariate membership functions

rithm based on a uniformly distributed training data where F_{NaOH} is in the range of 515-525 l/min. Our experiences show that 7 neurons are sufficient in the hidden layer of the NN. The results in Table 3.5.1 shows, that the neural network models give very good prediction performance for this process. Numbers in the brackets represent the number of neurons in the hidden layer and the removed neurons from the identified NN.

Testcase	**Training errors**(MSE)	**Testing Errors**(MSE)
Neural Network (7)	3.088e-005	3.267e-005
Using i-or (7)	3.053e-005	3.259e-005
Network reduction (8/1 neuron)	4.434e-005	4.285e-005
Network reduction (7/1 neuron)	3.060e-005	3.247e-005
Network reduction (6/2 neuron)	2.884e-004	3.690e-004
Network reduction (6/1 neuron)	1.086e-004	1.316e-004

Table 3.1: One-step ahead prediction results.

Applying the proposed visualization and transformation techniques, Fig. 3.6 shows the decomposed univariate membership functions and the into two dimension mapped distance matrix according to the NN model parameters can be seen on Fig. 3.7. For better interpretability, the histogram of the corresponding model inputs are illustrated on the last two subplots.

On Fig. 3.7 the pairwise distances of the neurons (see (3.8) in the previous section) were mapped into two dimensions with MDS and two dimensional points refer how similar the neurons behave.

The neurons are listed according to the OLS ranking on the left of Fig. 3.8, starting with the rules decomposed from the most important neuron in the hidden

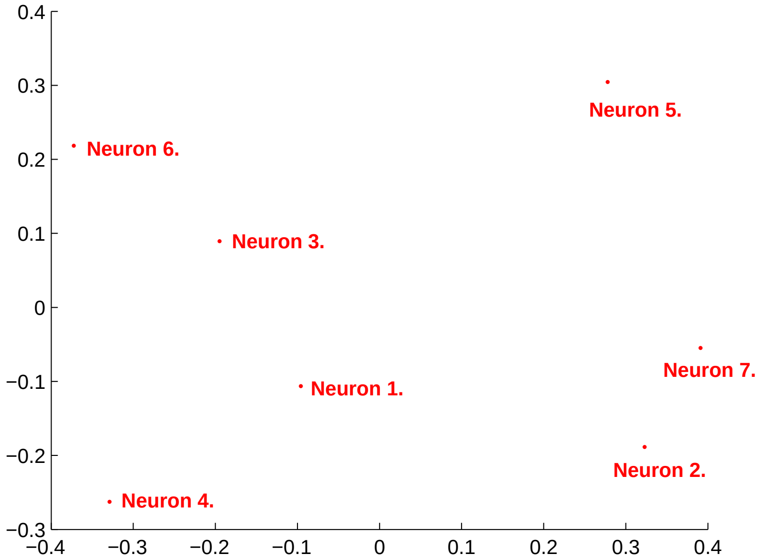

Figure 3.7: Distances between neurons mapped into two dimensions with mds

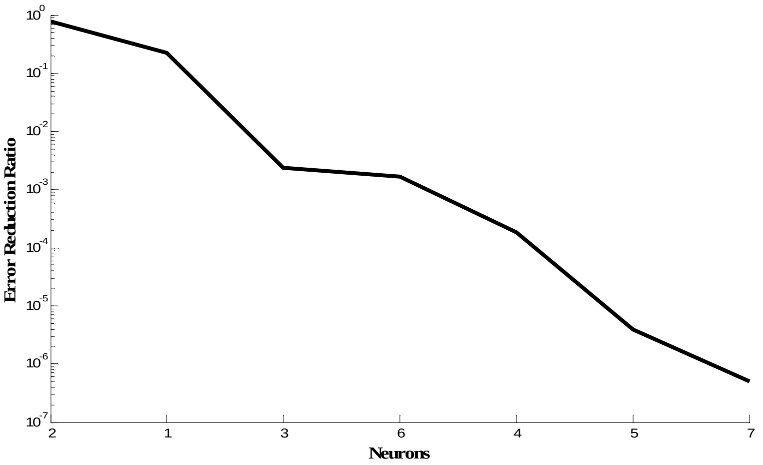

Figure 3.8: Error reduction ratios

network layer. The consequence of synthesizing the results is that it is possible to remove one neuron out of 7 (in FAS the corresponding rules) from the model without a significant increase in modeling performance, because of the low error reduction rate of the last, 7th neuron. This achievement harmonizes with the issues of the mapped distances, where the 2nd and the 7th neuron are closer to each other, but OLS based ranking indicates the 2nd one as more important.

Model reduction and visualization techniques like OLS makes it possible to overcome the problem of overfitting and the performance of the reduced model is almost the same as the original one. A rigorous test of NARX models is free run simulation because the errors can be cumulated. The result indicates the goodness of the reduced model even by free run simulation ($3.5 \cdot 10^{-3}$ for neural network with

–	6	7	8	9	10	11	12	13	14	15
1	$6.8 \cdot 10^{-4}$	$4.4 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$	$4.5 \cdot 10^{-5}$	$2.9 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	$6.6 \cdot 10^{-4}$	$3.6 \cdot 10^{-5}$	$6.6 \cdot 10^{-4}$
2	$4.8 \cdot 10^{-5}$	$6.7 \cdot 10^{-4}$	$3.9 \cdot 10^{-5}$	$3.9 \cdot 10^{-5}$	$6.7 \cdot 10^{-4}$	$3.4 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	$3.6 \cdot 10^{-5}$	$5.9 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$
3	$6.5 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$	$1.7 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	$3.1 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$	$8.9 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$
4	$6.4 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	$3.6 \cdot 10^{-5}$	$5.0 \cdot 10^{-5}$	$3.2 \cdot 10^{-5}$	$6.6 \cdot 10^{-4}$	$5.8 \cdot 10^{-5}$	$3.9 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$
5	$7.7 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$5.3 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	$4.2 \cdot 10^{-5}$
6	$--$	$2.9 \cdot 10^{-5}$	$4.4 \cdot 10^{-5}$	$5.6 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	$6.3 \cdot 10^{-4}$	$6.5 \cdot 10^{-4}$	$3.1 \cdot 10^{-5}$	$8.4 \cdot 10^{-5}$	$7.2 \cdot 10^{-5}$
7	$--$	$--$	$6.4 \cdot 10^{-4}$	$4.4 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	$6.2 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$	$4.4 \cdot 10^{-5}$
8	$--$	$--$	$--$	$1.7 \cdot 10^{-5}$	$6.1 \cdot 10^{-5}$	$7.9 \cdot 10^{-5}$	$3.7 \cdot 10^{-5}$	$3.2 \cdot 10^{-5}$	$3.6 \cdot 10^{-5}$	$7.1 \cdot 10^{-5}$
9	$--$	$--$	$--$	$--$	$4.8 \cdot 10^{-5}$	$4.4 \cdot 10^{-5}$	$1.7 \cdot 10^{-5}$	$4.9 \cdot 10^{-5}$	$3.7 \cdot 10^{-5}$	$4.7 \cdot 10^{-5}$
10	$--$	$--$	$--$	$--$	$--$	$3.2 \cdot 10^{-5}$	$3.2 \cdot 10^{-5}$	$8.7 \cdot 10^{-5}$	$9.0 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$
11	$--$	$--$	$--$	$--$	$--$	$--$	$3.1 \cdot 10^{-5}$	$7.0 \cdot 10^{-5}$	$5.0 \cdot 10^{-5}$	$1.5 \cdot 10^{-4}$
12	$--$	$--$	$--$	$--$	$--$	$--$	$--$	$3.6 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	$4.5 \cdot 10^{-5}$
13	$--$	$--$	$--$	$--$	$--$	$--$	$--$	$--$	$3.1 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$
14	$--$	$--$	$--$	$--$	$--$	$--$	$--$	$--$	$--$	$2.6 \cdot 10^{-5}$

Table 3.2: Training errors for different model structures and model reductions. (Number of neurons in the hidden layer of the network/Number of reduced neurons from the given model structure)

7 neurons, $3.823 \cdot 10^{-3}$ using i-or for FAS with 7 rules, $3.824 \cdot 10^{-3}$ after removing 1 neuron from the hidden layer containing 7 neurons).

In table 3.5.1 shows the training errors for neural networks with different number of neurons in the hidden layer $(6 - 15)$. These models were reduced with $1 - 14$ neurons. The obtained results points on that it is worth considering to select the appropriate model structure because better modeling performance can be achieved with reducing an overfitted model. The reason of this phenomena is that the gradient–based training algorithms may stop in different local minimums.

3.5.2 pH dependent structural relationship model for capillary zone electrophoresis of tripeptides

Aim of paper [80] was to study the structural descriptor–mobility relationship of representative tripeptides in capillary zone electrophoresis(CZE) separation length in respect to their influence on electrophoretic migration properties. For this purpose a back propagation neural network was applied with the inputs of *pH*, effective capillary length *l*, applied voltage *U*, peptide charge *O* and molecular weight *Mw*. 1000 iterations were used to train the neural network with the learning rate of 0.1. Number of nodes in the hidden layer was 15 and there was 1 output layer.

In the followings this data and model will be applied for introducing model reduction and visualization techniques.

Testcase	**Training errors**(MSE)	**Testing Errors**(MSE)
Neural Network (15)	3.088e-005	3.267e-005
Using i-or (15)	3.053e-005	3.259e-005
Network reduction (15/1 neuron)	4.434e-005	4.285e-005
Network reduction (16/1 neuron)	3.060e-005	3.247e-005
Network reduction (17/2 neuron)	2.884e-004	3.690e-004
Network reduction (18/3 neuron)	1.086e-004	1.316e-004

Table 3.3: One-step ahead prediction results.

The results in table 3.5.2 shows, that the originally utilized hidden layer with 15 neurons was carefully designed, because applying further reduction techniques on this structure cause decrease in model performance. However, it can be clearly stated as well that adding an extra neuron to the network and removing it can help in the model performance, without causing any overfitting of the neural network model.

3.6 Conclusions

Neural networks are often too complex and not interpretable, therefore it is very difficult to utilize these networks correctly. This article proposed a new complex approach for visualization and reduction of the neural networks, and discussed that neural network with sigmoid transfer function is identical to fuzzy additive systems.

The used similarity measure can be applied for further reduction of the rule base. It can be done in an automatic way if a threshold value is defined previously. If the measured similarity is greater than the threshold, the corresponding two neurons in the original neural network can be considered as identical; therefore further reduction of the FAS rule base is possible. This technique can be used even during the learning process of the neural network.

A possible future research area is to develop a new learning procedure for neural networks using prior knowledge based if-then rules, which combines the user's experience and/or constraints with the learning capability of NN.

Chapter 4

Support Vector Machines

Application of support vector methods for the initialization of fuzzy models is not a completely new idea. Numerous methods have been proposed to build the connection between the SVR and the FIS. Chen and Wang[22, 23] propose a positive definite fuzzy system (PDFS). In the proposed fuzzy model, the PDFS is equivalent to a Gaussian-kernel SVM[22] if Gaussian membership functions are adopted. Antecedent of a fuzzy rule is obtained by a support vector (SV). Therefore the number of fuzzy rules is the same as the number of SVs. As the number of SVs is generally large, the size of the FIS based on an SVM is also large. To solve this problem, researchers[81] proposed a learning algorithm to remove the irrelevant fuzzy rules. In spite of this, the generalization performance is degraded. The above methods are for zero-order FIS,which has one fuzzy singleton in the consequent of a fuzzy rule. For the first order FIS, Leski[25] describes a method for obtaining a FIS by means of the SVM with data-independent kernel matrix. Moreover, Juang et al. used a combination of fuzzy clustering and the linear SVM to establish a fuzzy model with less parameter number and better generalization performance. However, negligible effort has been done to establish a HFIS(high order FIS) with kernel methods. In [82] it was presented a HFIS with high accuracy and good generalization performance. It was shown how to obtain the formulation of the nonlinear function for the consequent part.

Furthermore, Catala used prototype vectors to combine with the support vectors using geometric methods to define ellipsoids in the input space, which are later transformed to if-then rules.[21]. In [83] special operator was utilized to achieve equivalency between support vector machines and fuzzy rule-based system. In [84] utilization of support vector models is described to solve the convex optimization problem for multivariate linear regression models and it is also shown how multivariate fuzzy nonlinear regression model can be formalized for numerical inputs and fuzzy output. Multiple types of kernels[84, 83] can be used to solve crisp nonlinear regression problems[85]. Chia et al. [86] used a combination of fuzzy clustering and linear support vector regression to obtain Takagi-Sugeno type fuzzy rules. Support vector machines can be applied to determine the support vectors for each fuzzy cluster obtained by fuzzy c-means clustering algorithm[87].

Visualization of fuzzy regression models is also discussed lately. Interpretation of fuzzy regression is provided with an insight into regression intervals so that regression interval analysis, data type analysis and variable selections is analytically performed[88]. In [89] a visualization and interpretation tool is presented. Feature space is visualized with highlighting the corresponding variables in the original input data to show how they are associated to the output variable. It is shown that which part of the input data can be utilized to estimate the output value. This technique also describes which input variable are responsible for the performance of the support vector regression. With the combination of visualization and interpretation the black-box support vector regression is identified in one step.

It must be taken into account that fuzzy logic does not guarantee interpretability as a prerequisite, because obtaining fuzzy models from support vector based training often result fuzzy models with high number of fuzzy rules. This phenomena makes interpretability much more difficult, therefore aim of this chapter is to describe a combination–of–tools three–step technique how to use reduction techniques on trained SVR models to acquire transparent, but accurate fuzzy rule based regression models. The steps are the following:

1. *Application of the Reduced Set method*

   The identification of the SVM is followed by the application of the Reduced Set (RS) method to decrease the number of kernel functions. Originally, this method has been introduced by [90] to reduce the computational complexity of SVMs. The obtained SVM is subsequently transformed into a fuzzy rule-based regression model.

2. *Similarity-based fuzzy set merging*

   The Gaussian membership functions of the fuzzy rule-based regression model are derived from the Gaussian kernel functions of the SVM. The interpretability of a fuzzy model highly depends on the distribution of the membership functions. Hence, the next reduction step is achieved by merging fuzzy sets based on a similarity measure [91].

3. *Rule-base simplification by orthogonal transformation*

   Finally, an orthogonal least-squares method is used to reduce the number of rules and re-estimate the consequent parameters of the regression model. The application of orthogonal transformations for reducing the number of rules has received much attention in the recent literature [92, 93]. These methods evaluate the output contribution of the rules to obtain the order of importance. The less important rules are then removed according this ranking to further reduce the complexity and increase the transparency.

This chapter organized as follows. Firstly basic notations of support vector machines and the connection between the fuzzy regression is described. After detailed description of the three-step reduction algorithm, examples indicating the power and the usage of described techniques on regression problems are presented.

4.1 FIS interpeted SVR

SVM has been recently introduced for solving pattern recognition and function estimation problems. SVM is a nonlinear generalization of the Generalized Portrait algorithm developed in Russia in the 1960s. In its present form, the SVM was developed at AT&T Bell Laboratories by Vapnik and co-workers[24]. SVM learning has now evolved into an active area of research. Moreover, the technique belongs to the standard methods toolbox of machine learning.

4.1.1 Support Vector Regression Models

The basic idea behind support vector regression is the kernel function: $k\left(\mathbf{x}_i, \mathbf{x}_j\right)$. Using k instead of dot product in $\mathbf{R}_i^N$, this will correspond to map the data into a possibly high dimensional space F, by a usually nonlinear map $\phi : \mathbf{R}_i^N \rightarrow F$ and take the dot product there

$$k\left(\mathbf{z}_i, \mathbf{x}\right) = \left(\phi\left(\mathbf{z}_i\right), \phi\left(\mathbf{x}\right)\right) \tag{4.1}$$

SVR concept will be introduced based on [94], for more detail please see [17]. Suppose we have training data $\left\{\left(x_1, y_1\right), \ldots, \left(x_{N_d}, y_{N_d}\right)\right\} \subset \chi \times \mathbf{R}_i^N$, where χ denotes the space of input patterns. The aim is to find function $f\left(x\right)$ that has at most ε deviation from the targets with the obtained y_i, for all the training data. In other words we do not care about the errors as long as they are less than ε, but any larger deviation than ε won't be accepted. SVR can be formulated as follows:

$$\begin{aligned} &\min_{w,b,\xi_i,\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N_d}(\xi_i + \xi_i^*) \\ &s.t.\ \ y_i - \mathbf{w}^T\phi(\mathbf{x}_i) - b \leq \varepsilon + \xi_i \\ &\qquad \mathbf{w}^T\phi(\mathbf{x}_i) + b \leq \varepsilon + \xi_i^* - y_i \\ &\qquad\qquad\qquad\qquad \xi_i, \xi_i^* \geq 0 \end{aligned} \tag{4.2}$$

where ϕ is the feature mapping for kernel k, ε is the tolerance error, ξ_i, ξ_i^* are slack variables and $C > 0$ is a cost coefficient, which determines the trade-off between the model complexity and the degree of tolerance to the errors larger than ε. The dual form of the optimization problem 4.2 becomes a quadratic programming (QP) problem:

$$\begin{aligned} &\max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i,j=1}^{N_d}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)k(\mathbf{x}_i, \mathbf{x}_j) \\ &\qquad -\varepsilon\sum_{i=1}^{N_d}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N_d} y_i(\alpha_i - \alpha_i^*) \\ &s.t.\ \sum_{i=1}^{N_d} y_i(\alpha_i - \alpha_i^*) = 0 \ \ \alpha, \alpha^* \in [0, C] \end{aligned} \tag{4.3}$$

where α and α^* are the Lagrange multipliers. As an outcome of solving the QP problem 4.3 can be rewritten to the following form:

$$f(x) = \sum_{i=1}^{N_d}(\alpha_i - \alpha_i^*)k(\mathbf{x}_i, \mathbf{x}) + b \tag{4.4}$$

Let $\gamma_i = \alpha_i - \alpha_i^*$. In case $\gamma_i \neq 0$ the corresponding training pattern x_i can be noted as support vector.

4.1.2 Structure of Fuzzy Rule-based regression model

To get a fuzzy-rule based regression model from the support vector regression model the following interpretation is needed:

$$y = \sum_{i=1}^{N_R} \beta_i(\mathbf{x})\delta_i + b, \tag{4.5}$$

where β_i is the firing strength and δ_i is the rule consequent. The output of the regression model is calculated by this equation. In case of fuzzy systems, fuzzy rules can be formulated as follows

$$R_i\, if \;\; x_1 \; is \; A_{i1} \; and \ldots x_n \; is \; A_{in} \; then \; y_i = \delta_i, \quad i = 1, \ldots, N_R \;, \tag{4.6}$$

where R_i is the ith rule in the fuzzy rule-based regressor and N_R denotes the number of rules. $A_i, \ldots, A_{N_i}$ denote the antecedent fuzzy sets that define operating region of rule in the N_i dimensional input space. The rule consequent δ_i is a crisp number. The connective is modeled by the product operator. Hence the degree of activation of the ith rule is calculated as

$$\beta_i(\mathbf{x}) = \prod_{j=1}^{N_i} A_{ij}(\mathbf{x}_j), \;\; i = 1, \ldots, N_R. \tag{4.7}$$

Main principle of kernel-based support vector regressors is the identification of a linear decision boundary in this high dimensional feature-space. The link to the fuzzy model structure is the following: The fuzzy sets are represented by Gaussian membership functions

$$A_{ij}(\mathbf{x}_j) = exp\left(\frac{(\mathbf{x}_j - \mathbf{z}_{ij})^2}{2\sigma^2}\right) \tag{4.8}$$

The degree of fulfilment $\beta_i(\mathbf{x})$ can be written through 4.7-4.8 in a more compact form by using Gassuian kernels.

$$\beta_i(\mathbf{x}) = exp\left(\frac{\|\mathbf{x} - \mathbf{z}_j\|^2}{2\sigma^2}\right) \tag{4.9}$$

This kernel interpretation of fuzzy systems shows that fuzzy models are effective in solving nonlinear problems because they map the original input space into a nonlinear feature space by using membership functions similarly to the support vector machine that utilize kernel functions for this purpose.

4.2 Ensuring interpretability with three–step algorithm

In the previous sections it has been shown how a SVM, that is structurally equivalent to a fuzzy model, can be identified. Unfortunately, this identification method cannot be used directly for the identification of interpretable fuzzy systems because the number of the support vectors is usually very large. Typical values are 40-60$\%$ of the number of training data which is in our approach equal to the number of rules in the fuzzy system. Therefore, there is a need for an interpretable approximation of the support vector expansion. For this purpose a step-wise algorithm will be introduced, where the first step is based on the recently published Reduced Set (RS) method developed for reducing the computational demand of the evaluation of SVMs [90].

4.2.1 Model Simplification by Reduced Set Method

The aim of the RS method is to approximate the high-dimensional feature space given by the support vectors

$$\Psi = \sum_{i=1}^{N_x} \gamma_i \phi(\mathbf{x}_i) \ , \tag{4.10}$$

by a reduced set expansion

$$\Psi' = \sum_{i=1}^{N_R} \delta_i \phi(\mathbf{z}_i) \ , \tag{4.11}$$

with $N_R < N_x < N_d$, where N_x denotes the number of support vectors (the number of $\mathbf{x}_i$ vectors for those $\gamma_i \neq 0$) and N_R represents the number of the desired rules in the fuzzy rule-based regressor that we would like to identify and $\mathbf{z}_i$ denotes the centers of the new kernel functions that are not necessarily training samples. N_R should be as small as possible because it determines the number of fuzzy rules. In practice it turns out that the RS method is often able to deliver a one-tenth reduction, so N_R can be chosen as $N_R = N_x/10$. For this model reduction, the squared error $||\Psi - \Psi'||^2$ has to be minimized. For this purpose, the 'kernel trick' has to be applied because ϕ is not given explicitly

$$\begin{aligned} \|\Psi - \Psi'\|^2 &= \sum_{i,j=1}^{N_x} \gamma_i \gamma_j k(\mathbf{x}_i, \mathbf{x}_j) + \\ & \sum_{i,j=1}^{N_R} \delta_i \delta_j k(\mathbf{z}_i, \mathbf{z}_j) - 2 \sum_{i}^{N_x} \sum_{j}^{N_R} \gamma_i \delta_j k(\mathbf{x}_i, \mathbf{z}_j) \ . \end{aligned} \tag{4.12}$$

The cost function 4.12 is minimized in a step-wise manner while the feature space is approximated by the following iterative algorithm:

Repeat for $m : 2, \ldots, N_R$;

- **Step 1:***Obtain the residual space*

  Let Ψ_m mean the residual of the feature space approximation generated at the $(m$-1$)$–th step

  $$\begin{aligned} \Psi_m &= \sum_{i=1}^{N_x} \gamma_i \phi(\mathbf{x}_i) - \sum_{i=1}^{m-1} \delta_i \phi(\mathbf{z}_i) \\ &= \sum_{i=1}^{N_m} \epsilon_i \phi(\mathbf{v}_i) \ , \end{aligned} \tag{4.13}$$

  where

  $$\begin{aligned} &(\epsilon_1, \ldots, \epsilon_{N_m}) = (\gamma_1, \ldots, \gamma_{N_x}, -\delta_1, \ldots, -\delta_{m-1}) \ , \\ &(\mathbf{v}_1, \ldots, \mathbf{v}_{N_m}) = (\mathbf{x}_1, \ldots, \mathbf{x}_{N_x}, \mathbf{z}_1, \ldots, \mathbf{z}_{m-1}) \ , \\ &N_m = N_x + m - 1 \ . \end{aligned}$$

- **Step 2:***Inner iteration step for determining* $\mathbf{z}_m$

  This residual function is approximated by the determination of $\mathbf{z}_m$ and δ_m in the iterative procedure, where the following cost function has to be minimized

  $$\min_{\delta_m, \mathbf{z}_m} \|\Psi_m - \delta_m \phi(\mathbf{z}_m)\|^2 \ . \tag{4.14}$$

  This can be done by standard techniques or using fixed–point iteration, as shown in [90].

$$\mathbf{z}_m^{n+1} = \frac{\sum_{i=1}^{N_m} \epsilon_i \exp\left(-\|\mathbf{v}_i - \mathbf{z}_m^n\|^2 / \left(2\sigma^2\right)\right) \mathbf{v}_i}{\sum_{i=1}^{N_m} \epsilon_i \exp\left(-\|\mathbf{v}_i - \mathbf{z}_m^n\|^2 / \left(2\sigma^2\right)\right)}\ , \tag{4.15}$$

where the superscript n denotes the n–th inner iteration step; 4.15 is iterated till it converges to $\|\mathbf{z}_m^{n+1} - \mathbf{z}_m^n\|^2 < \epsilon$.

Interestingly, 4.15 can be interpreted in the context of clustering [95]. It determines the center of a single Gaussian cluster, trying to capture as many of the $\mathbf{v}_i$ with positive δ_i as possible, and simultaneously avoiding those $\mathbf{v}_i$ with negative δ_i.

- **Step 3:***Least-squares estimation of the δ_i coefficients*

  The δ_m coefficient is calculated by recalculating the whole δ=$[\delta_1, \dots, \delta_m]^T$ vector by minimizing 4.12

$$\delta = (K^z)^{-1} K^{zx} \gamma \tag{4.16}$$

  where the element of the matrices are expressed by the kernel functions $K_{ij}^z = k(\mathbf{z}_i, \mathbf{z}_j)$ and $K_{ij}^{zx} = k(\mathbf{z}_i, \mathbf{x}_j)$.

4.2.2 Reducing the Number of Fuzzy Sets

In the previous section, it has been shown how kernel-based regression model with a given number of kernel functions N_R, can be obtained. Because the number of rules in the transformed fuzzy system is identical to the number of kernels, it is extremely important to get a moderate number of kernels in order to obtain a compact fuzzy rule-based regression model.

From Eq. 4.9 it can be seen that the number of fuzzy sets in the identified model is $N_s = N_R N_i$. The interpretability of a fuzzy model highly depends on the distribution of these membership functions. With the simple use of Eq. 4.8, some of the membership functions may appear almost undistinguishable. Merging similar fuzzy sets reduces the number of linguistic terms used in the model and thereby increases model transparency. This reduction is achieved by a rule-base simplification method [91, 96], based on a similarity measure $S(A_{ij}, A_{kj})$, $i, k = 1, \dots, n$ and $i \neq j$. If $S(A_{ij}, A_{kj}) = 1$, then the two membership functions A_{ij} and A_{kj} are equal. $S(A_{ij}, A_{kj})$ becomes 0 when the membership functions are nonoverlapping. During the rule-base simplification procedure similar fuzzy sets are merged when their similarity exceeds a user-defined threshold $\theta \in [0, 1]$. The set-similarity measure can be based on the set-theoretic operations of intersection and union [91].

$$S(A_{ij}, A_{kj}) = \frac{|A_{ij} \cap A_{kj}|}{|A_{ij} \cup A_{kj}|}\,, \tag{4.17}$$

where $|.|$ denotes the cardinality of a set, and the $\cap$ and $\cup$ operators represent the intersection and union, respectively, or it can be based on the distance of the two fuzzy sets. Here, the following expression was used to approximate the similarity between two Gaussian fuzzy sets [96]

$$\begin{aligned} S(A_{ij}, A_{kj}) &= \frac{1}{1 + d(A_{ij}, A_{kj})} \qquad (4.18)\\ &= \frac{1}{1 + \sqrt{(z_{ij} - z_{kj})^2 + (\sigma_{ij} - \sigma_{kj})^2}}\,. \end{aligned}$$

4.2.3 Reducing the Number of Rules by Orthogonal Transforms

By using the previously presented SVM identification and reduction techniques, the following fuzzy rule-based regression model has been identified

$$y = \sum_{i=1}^{N_R} \prod_{j=1}^{N_i} \exp\left(\frac{(x_j - z_{ij})^2}{2\sigma^2}\right) \delta_i + b. \tag{4.19}$$

Due to the applied RS method and the fuzzy set merging procedure, the obtained membership functions only approximate the original feature space identified by the SVM. Hence, the $\delta=[\delta_1, \ldots, \delta_{N_r}]^T$ consequent parameters of the rules have to be re-identified to minimize the difference between the decision function of the support vector machine Eq. 4.4 and the fuzzy model Eq. 4.19

$$\begin{aligned} MSE &= \sum_{j=1}^{N_d} \left(\sum_{i=1}^{N_x} \gamma_i k(\mathbf{x}_j, \mathbf{x}_i) - \sum_{i=1}^{N_R} \delta_i \beta_i(\mathbf{x}_j) \right)^2 \qquad (4.20)\\ &= \|\mathbf{y}_s - \mathbf{B}\delta\|^2\,, \qquad (4.21) \end{aligned}$$

where the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_{N_R}] \in \mathbf{R}^{N_d \times N_R}$ contains the firing strength of all N_R rules for all the inputs $\mathbf{x}_i$, where $\mathbf{b}_j = [\beta_j(\mathbf{x}_1), \ldots, \beta_j(\mathbf{x}_{N_d})]^T$. As the fuzzy rule-based regression model 4.19 is linear in the parameters δ, Eq. 4.20 can be solved by a least-squares method(see Appendix C and Eq. C.1 for further details).

The application of orthogonal transformations for the above mentioned regression problem Eq. 4.20 for reducing the number of rules has received much attention in recent literature [92, 93].

For modeling purposes, the Orthogonal Least Squares (OLS) is the most appropriate tool [92]. The OLS method transforms the columns of $\mathbf{B}$ into a set of orthogonal basis vectors in order to inspect the individual contribution of each rule. This ratio offers a simple mean for ordering the rules, and can be easily used to select a subset of rules in a forward-regression manner.

Evaluating only the approximation capabilities of the rules, the OLS method often assigns high importance to a set of redundant or correlated rules. To avoid this, in [93, 97] some extension for the OLS method were proposed.

4.3 Application Examples

4.3.1 Illustrative example

To demonstrate the potential of Support Vector Regression techniques two examples were introduced. Firstly, an illustrative regression problem is solved with a simple dataset containing 51 samples (Fig.4.1). The SVR technique obtained 14 support vectors. This model has been reduced by the RS method (Step 1.), by which we tried to reduce the model to operate with 10 rules. Modeling results can be seen in Table 4.1. Utilization of all the three algorithm steps reduced the number of fuzzy rules to 6, however this indicated slight increase in modeling error.

Table 4.1: Results on Regress data

Method	RMSE	#Rules
SVR Identification	0.084	14
Step 1 reduction	0.0919	10
Step 2 reduction	0.2415	9
Step 3 reduction	0.3361	6

4.3.2 Identification of Hammerstein System

In this example, the support vector regression is used to approximate a Hammerstein system that consists of a series connection of a memory less nonlinearity, f, and linear dynamics, G, as shown in Fig. 4.2 where v represents the transformed input variable. For transparent representation the Hammerstein system consist of a first-order linear part $y\,(k+1) = 0,9y(k) + 0,1v(k)$ and a static nonlinearity is

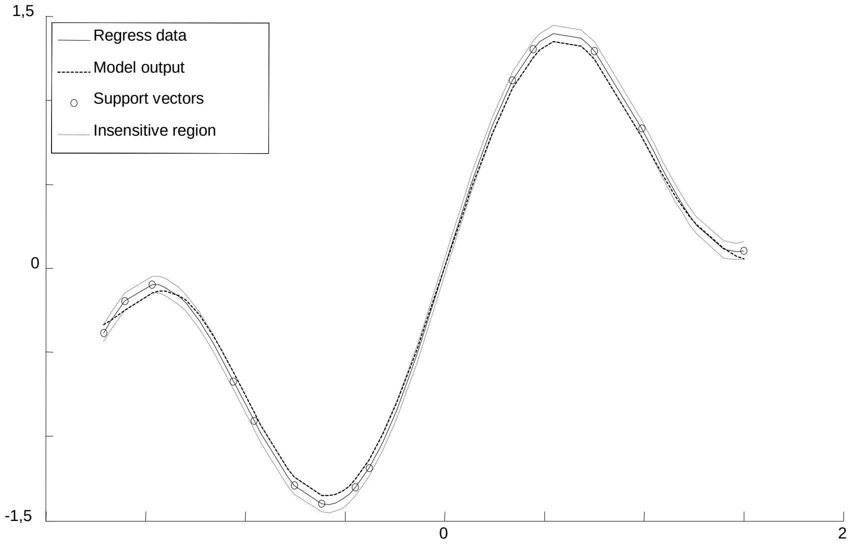

Figure 4.1: Illustrative example with model output, support vectors and the insensitive region

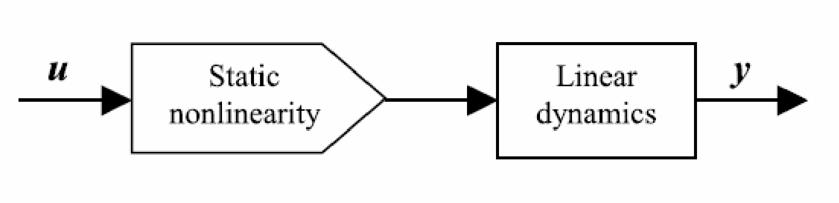

Figure 4.2: Hammerstein system

represented by a polynomial, $v(k) = u(k)^2$. The dataset contains 500 input-output data. Support vector regression model was identified with efficiency summarized in Table 4.2.

Table 4.2: Results on Hammerstein system identification

Method	RMSE	#Rules
SVR identification	0.0533	22
Step 1 reduction	0.0604	15
Step 2 reduction	0.0650	13
Step 3 reduction	0.0792	12

As Fig. 4.3 and Table 4.2 concludes, support vector regression is able to give accurate models for Hammerstein system identification. Extracted, non-distinguishable rules from this system are represented on Fig. 4.4, therefore the three-step reduction algorithm is used to acquire interpretable models.

After applying the RS method (Step 1.), number of rules could be reduced to 15 without any major modeling error increase. Using further reductions with the

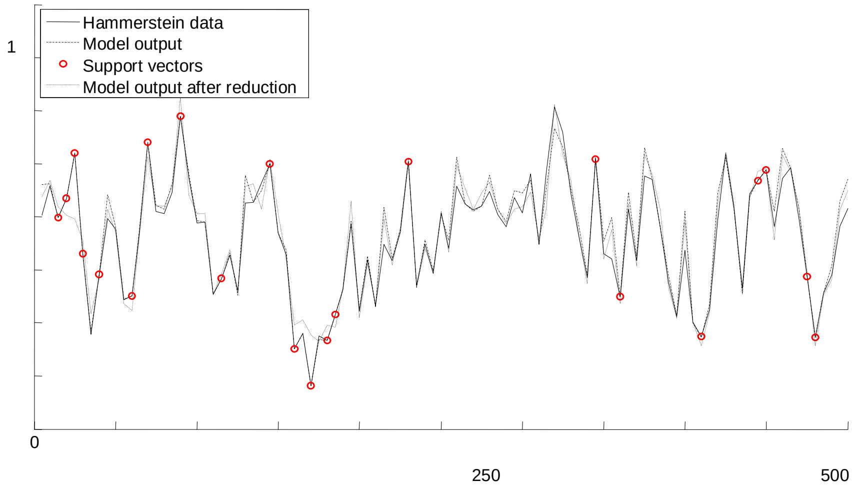

Figure 4.3: Identified Hammerstein system, support vectors and model output after reduction

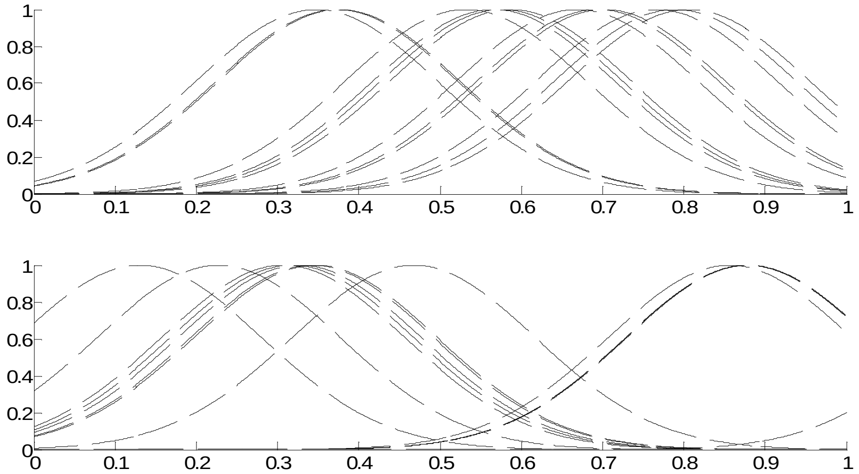

Figure 4.4: Non-distinguishable membership functions obtained after the application of RS method

second and third step of the proposed algorithm, interpretable model (see Fig. 4.5) and accurate model could be extracted.

4.4 Conclusions

Support vector based techniques and fuzzy rule-based models work in a similar manner as both models maps the input space of the problem into a feature space with the use of either nonlinear kernel or membership functions. The main difference between support vector based and fuzzy rule-based systems is that fuzzy systems have to fulfil two objectives simultaneously, i.e., they must provide a good modeling performance and must also be linguistically interpretable, which is not an issue for

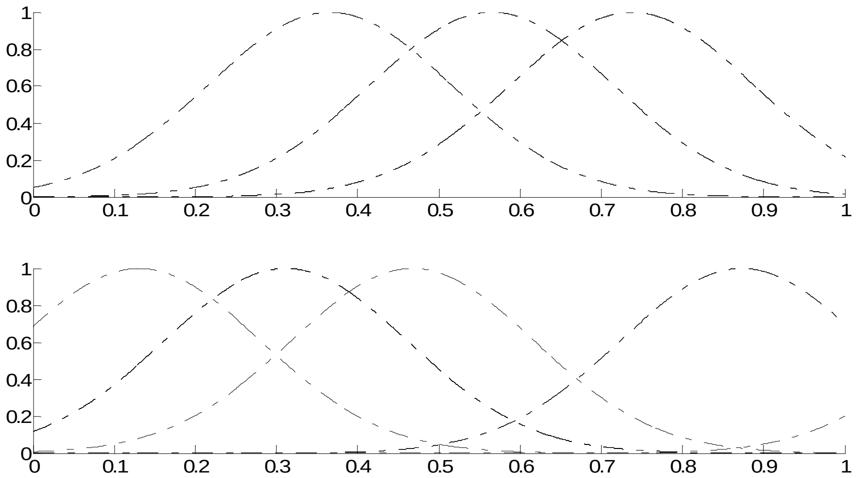

Figure 4.5: Interpetable membership functions of the reduced fuzzy model

support vector systems. However, as the structure identification of fuzzy systems is a challenging task, the application of kernel-based methods for model initialization could be advantageous because of the high performance and the good generalization properties of these type of models.

Accordingly, support vector-based initialization of fuzzy rule-based model is used. First, the initial fuzzy model is derived by means of the support vector learning algorithm. Then the support vector model is transformed into an initial fuzzy model that is subsequently reduced by means of the reduced set method, similarity-based fuzzy set merging, and orthogonal transform-based rule-reduction. Because these rule-base simplification steps do not utilize any nonlinear optimization tools, it is computationally cheap and easy to implement them. The application of the proposed approach was shown on simple one–dimensional function identification data and Hammerstein system identification. The obtained models are very compact but their accuracy is still adequate. Besides, it might be clear that still real progress can be made in the development of novel methods for feature selection.

I intend this thesis also as a case study for further developments in the direction of a combination-of-tools methodology for modeling and identification. I am seeking for techniques that perform well on multiple criteria, considering here different soft-computing tools combined to achieve a predefined trade-off between performance and transparency.

Chapter 5

Summary

5.1 Introduction

Majority of problems arisen in chemical engineering practice requires data-driven modeling of nonlinear relationships between experimental and technological variables. Complexity of nonlinear regression techniques is gradually expanding with the development of analytical and experimental techniques, hence model structure and parameter identification is a current and important topic in the field of nonlinear regression not just by scientific but also from industrial point of view as well. Model interpretability is the most important key property besides accuracy in the regression modeling of technological processes and this is essential characteristic of these models in their application as process controllers. As it was mentioned above, model structure and parameter identification is an actual topic with increasing importance, since identified model needs to be interpretable as well. In line with these expectations and taking interpretability of regression models as basic requirement robust nonlinear regression identification algorithms were developed in this thesis. Three algorithms were examined in details namely identification of regression trees based hinging hyperplanes, neural networks and support vector regression. Application of these techniques eventuate black box models at first step. It is shown in my thesis how interpretability could be maintained during model identification with utilization of applicable visualization and model structure reduction techniques within the fuzzy modeling framework.

First part of the thesis deals with the identification of hinging hyperplanes based regression trees. Results of the developed algorithm prove that the implementation of a priori constraints enables fuzzy c-regression clustering technique to identify hinging hyperplane models. Application of this technique recursively on the partitioned input space ends up in a regression tree capable for modeling and even for implementation of model predictive control of technological data coming from real life applications. The next section deals with the validation, visualization and structural reduction of neural networks. It is described in details that the hidden layer of the neural network can be transformed to an additive fuzzy rule base.

This section is followed by the description of connections between fuzzy regression and support vector regression, and introduces a three-step reduction algorithm to get interpretable fuzzy regression models on the basis of support vector regression.

5.2 New Scientific Results

1. *I showed that hinging hyperplane models are excellent tools for the identification of models based on technological data. I tailored a new model structure by the hierarchical representation of hinging hyperplane models and I delivered a new identification algorithm based on fuzzy clustering.*

a) To overcome the problems of original hinge hyperplane identification algorithm delivered by Breimann [1] I adapted a fuzzy c-regression clustering algorithm for hinge identification with incorporating a priori constraints.

b) As further enhancement of this algorithm I developed hierarchical hinge hyperplane based on regression tree identification technique. I showed performance of the developed tool on multiple examples from the well–know repositories.

c) I proved that the identified transparent and interpretable models - with the help of the developed algorithm - are suitable for solving process control duties of technological systems. To illustrate this feature I presented model predictive control of a simulated cartridge water heater.

(Relevant publications: 1,7,9,10,11,13,14,18,19,20)

2. *To reinforce support vector regression methods, I worked out a three step reduction technique in order to reduce and transform the support vector model into an interpretable fuzzy rule base. Further reduction of this rule base implies* *interpretable and robust regression models.*

a) I examined structural equivalency between support vector and fuzzy regression. I worked out a technique with the help of Gaussian kernels to transform the identified support vector model into a fuzzy rule base.

b) Based on the identified support vector regression model, the transformed fuzzy rule base generates large number of rules making the model interpretation and validation difficult. I tailored a three step reduction algorithm to overcome this problem. I used the reduced set method [2] to select the important set of support vectors and I utilized further, similarity based reduction of the generated rule base. The resulted fuzzy rule base is linear in the consequent part, therefore I applied orthogonal least squares algorithm for further reduction.

(Relevant publications: 2, 5, 6)

3. *Interpretability of neural network models can be achieved by transforming hidden layer of the neural network into a fuzzy rule base and with using a special, self–developed visualization technique of this rule base. Based on the self–developed transformation and visualization technique I reduced the generated model with orthogonal least squares and similarity based reduction techniques in order to support proper model structure design.*

a) I examined that validation and interpretability of black box neural network models can be improved by transforming the hidden layer of the neural network models with a special operator to a fuzzy rule base.

b) I compared calculated membership functions based on similarity measure enabling the analysis of the neural network model and point out possible further model reductions. This model structure is also linear in parameters from the output layer point, so I used the mentioned orthogonal least squares technique for further model reduction.

c) Visualization of the neurons taking place on the hidden layer of neural network can be achieved by distance measure and multi-dimensional scaling. This technique is also a new tool to examine and validate structure of the neural network. Performance of these techniques is shown on a technological pH process.

(Relevant publications: 3, 4, 12, 15, 16, 17, 19)

5.3 Utilization of Results

The motivation to write my thesis was to integrate data, prior knowledge and extracted information into a single framework that helps model-building procedures with interpretability, visualization and reduction. Utilization of the developed algorithms was shown by section-wise examples taken from the area of chemical engineering. Benchmarks and experimental data were used to perform a most comprehensive test of novel methods.

Due to computational efficiency and easy interpretation, the hierarchical representation of hinging hyperplane model proved to be a promising tool to develop local linear controllers. Interpretable fuzzy regression models initialized by robust support vector regression could help when besides quantitative relationships, qualitative analysis is needed as well. The interpretable property of fuzzy models is a great vehicle for variable quality characterization. Structural validation and visualization of neural network models can support modellers to solve the challenge in case only black box model identification is possible. My self-developed technique gives excellent feedback to determine model structure and evaluate task complexity. In the chemical industry, these problems occur when trying to find connections between complex reaction kinetic relationships and key technology- and product-quality variables.

Future developments of the thesis at hand can branch in various directions in the field of interactive learning where modeller experience combined with learning capability of different identification techniques can lead to further successes.

Összefoglaló

5.4. Bevezetés

A vegyészmérnöki gyakorlatban elõforduló problémák jelentõs része kísérleti és technológiai változók közötti nemlineáris összefüggések adat-alapú modellezését követeli meg. A nemlineáris regressziós problémák komplexitása a technológiák és a kísérleti, analitikai technikák fejlõdésével folyamatosan növekszik, így a nemlineáris regresszió kapcsán a modellstruktúra meghatározása és a modell paramétereinek identifikálása tudományos és ipari szempontból is egyre fontosabb problémakör. A regressziós problémák egyre növekvõ komplexitása miatt tudományos és mûszaki szempontból is fontos és aktuális problémakör a modellstruktúra, illetve a modell paramétereinek meghatározása. Az identifikált modellekkel szemben a pontosságon túl a modell értelmezhetõsége a legfontosabb ismérv, mely a regressziós modellek technológiai folyamatok modellezésében és szabályozásában történõ alkalmazhatóságnak is fontos alapja. Ezen megállapítások kapcsán a regressziós modellek értelmezhetõségét szem elõtt tartva robosztus nemlineáris regressziós modellezési technikákat fejlesztettem. A vizsgált metszõ hipersík alapú regressziós fákon, neurális hálózatokon és szupport vektor regresszión alapuló modellezési technikák alapvetõen fekete doboz modelleket identifikálnak.

Értekezésemben megmutattam, hogy a modellek identifikációja során miként javítható a modellek értelmezhetõsége a megfelelõ megjelenítési és struktúra redukálási technikák segítségével, illetve a fuzzy modellezés keretrendszerének alkalmazásával.

Az értekezés elsõ részében metszõ hípersik alapú modellek identifikációjával foglalkozom. A kifejlesztett algoritmus eredményei igazolják, hogy az a prori korlátokon alapuló fuzzy c-regressziós csoportosítás technika alkalmas metszõ hipersík modellek identifikációjára, illetve a rekurzívan alkalmazott, regressziós fába rendezett hierarchikus modellstruktúra alkalmas technológiai folyamat modellezésre és modell prediktív szabályzás implementálására. Az értekezés következõ fejezete a neurális hálózatok validálásával, a hálózat rejtett rétegének fuzzy additív szabálybázissá való átalakításával, az eredmények transzparens ábrázolásával és a neurális hálózat redukciójával illetve vizualizálásával foglalkozik, bemutatván az alkalmazott technikák erõsségeit.

A dolgozat záró fejezete a szupport vektor regresszió által identifikált modell és a fuzzy regresszió közötti összefüggésekkel, a kapott modellstruktúra háromlépcsõs redukciós algoritmusával foglalkozik, illetve mutatja be annak mûködését és eredményességét.

5.5. Új tudományos eredmények

1. *Kimutattam, hogy a metszõ hipersík modellek kitûnõ eszközök technológiai adatok alapján történõ modellalkotásra. A metszõ hipersíkok hierarchikus modellbe történõ szervezésével egy új modellstruktúrát alkottam. E modellek identifikációjára fuzzy csoportosításon alapuló technikát dolgoztam ki.*

a) Breimann [1] által elsõként publikált metszõ hipersík modellek identifikációjára ajánlott algoritmus hibáinak kiküszöbölésére a fuzzy c-regressziós csoportosítás (fuzzy c-regression clustering) algoritmust a priori korlátok beépítésével alkalmassá tettem metszõ hipersík modellek identifikálására.

b) Az algoritmus továbbfejlesztésével egy hierarchikus metszõ hipersík alapú regressziós fa identifikációs technikát készítettem. Az algoritmus képességeit több, az irodalomban gyakorta alkalmazott példán keresztül bemutattam.

c) Igazoltam, hogy az elkészített algoritmus segítségével alkotott transzparens és értelmezhetõ modellek könnyen alkalmazhatóak technológiai rendszerek szabályzási feladatainak elvégzésére. Ennek illusztrálására egy valós vízmelegítõ dinamikus szimulátorának modell prediktív szabályozását valósítottam meg.

(Kapcsolódó publikációk: 1,7,9,10,11,13,14,18,19,20)

2. *Szupport vektor gépek regressziós feladatainak támogatása érdekében olyan módszert dolgoztam ki, amellyel a kapott szupport vektor modell redukálható, illetve átalakítható értelmezhetõ fuzzy szabálybázissá, és ezen szabálybázis további redukciójával transzparens, ugyanakkor robosztus modellek nyerhetõek.*

a) Megvizsgáltam a strukturális ekvivalencia kérdését szupport vektor regresszió és a fuzzy regresszió között. Olyan módszert dolgoztam ki, amellyel a Gauss kernel függvények használatával az identifikált szupport vektor modell fuzzy szabálybázissá alakítható.

b) Az identifikált szupport vektor regressziós modellben kapott szupport vektorok függvényében a fuzzy modell általában az értelmezhetõséget megnehezítõ nagy számosságú szabályt generál, ami megnehezíti az eredmények interpretálását és validálását ezért három lépcsõs redukciós algoritmust készítettem. A redukált halmazok módszerét [2] alkalmazva a lényeges szupport vektorok kiválasztására, majd az így kapott szabálybázist tovább csökkentettem hasonlósági mérték segítségével. Mivel az így kapott modell a fuzzy szabálybázis következmény paramétereire nézve lineáris, ezért az ortogonális legkisebb négyzetek módszerének segítségével tovább redukáltam a szabályok számát.

(Kapcsolódó publikációk: 2, 5, 6)

3. *A neurális hálózati modellek értelmezhetõsége megvalósítható a modellek fuzzy szabálybázissá történõ transzformációjával és a fuzzy modell szabályainak általam kidolgozott speciális megjelenítésével. Az általam kidolgozott transzformációs és megjelenítési technikán alapulva a kapott modellen ortogonális legkisebb négyzetek módszere és a hasonlósági mértéken alapuló redukciós technikák alkalmazhatók a modellstruktúra tervezésének támogatása érdekében.*

a) Igazoltam, hogy a fekete doboz neurális hálózatok értelmezhetõsége és validálása miképp javítható a neurális hálózati modellek rejtett rétegének egy speciális operátoron alapuló fuzzy additív szabálybázissá transzformálásával.

b) A kapott tagsági függvényeken értelmezett hasonlósági mértékek alapján lehetõség nyílik a kapott neurális hálózat elemzésére, redukálására. Tekintettel arra, hogy a modell a kimeneti réteg paramétereire nézve lineáris, a további modell redukció érdekében ortogonális legkisebb négyzetek technika használatát javasoltam.

c) A kapott neurális hálózat rejtett rétegének neuronjait távolság mérték alapján a többdimenziós skálázási technika segítségével vizualizáltam, amellyel egy újabb eszközt mutattam be a modell rejtett struktúrájának feltárására és validálására. A technikák képességeit technológiai adatokon, egy pH szabályozási folyamaton keresztül illusztráltam.

(Kapcsolódó publikációk: 3, 4, 12, 15, 16, 17, 19)

5.6. Az eredmények gyakorlati hasznosítása

A dolgozat motivációja, hogy olyan eszközt adjon modell építési folyamatok támogatására, ahol az adat, a-priori ismeret és a kinyert információ egységes keretrendszert alkot. Az eszközök segítségével értelmezhetõ, vizualizálható és redukálható modelleket kaphatunk. A fejlesztett algoritmusok használatát az egyes fejezetekben gyakorlati példákkal illusztráltam. A lehetõ legrészletesebb tesztelés megvalósításának érdekében kísérleti és mesterséges adatokat is felhasználtam a kidolgozott módszerek hatékonyságának bemutatására.

A metszõ hipersíkok számítási hatékonysága és könnyû értelmezhetõsége miatt ezek a hierarchikus modellek ígéretes eszközei lehetnek a lokálisan lineáris szabályozók fejlesztésének. A robosztus szupport vektor regresszió segítségével inicializált értelmezhetõ fuzzy regressziós modell jól használhatóak abban az esetben, amikor a kvantitatív összefüggések definiálásán kívül kvalitatív elemzés is szükséges. E témakörben a fuzzy modellek értelmezhetõsége a változók minõségi jellemzésében bizonyulhat különösen hasznos tulajdonságnak. A neurális hálózatok struktúra validálása és vizualizációja a modellezõt tudja támogatni olyan kihívások megoldásában ahol csupán fekete doboz modellek identifikálására van csak lehetõség. Az általam kidolgozott módszertan kitûnõ visszajelzést ad a modell struktúrájának meghatározásához és a modellezési feladat komplexitásának jellemzéséhez.

A vegyipari gyakorlatban ilyen jellegû problémák általában a komplex reakció kinetikai összefüggések és a technológiai és termékminõséget jellemzõ változók közti kapcsolatok feltárása során fordulnak elõ.

A dolgozat jövõbeni fejlesztése különféle irányokba tud elágazni az interaktív tanulás terén ahol a modellezõ tapasztalata és az identifikációs technikák tanulási képességeinek kombinációjával érhetünk el további sikereket.

Publications related to theses

Articles in International and Hungarian Journals

1. T. Kenesei, B. Feil, J. Abonyi, Fuzzy Clustering for the Identification of Hinging Hyperplanes Based Regression Trees *Lecture notes in computer science, Lecture notes in artificial intelligence; 4578. ISBN:9783540733997* pp. 179-186. 2007

2. T. Kenesei, A. Roubos, J. Abonyi, A Combination-of-Tools Method for Learning Interpretable Fuzzy Rule-Based Classifiers from Support Vector Machines *Lecture Notes in Computer Science; 4881. ISBN:978-3-540-77225-5* pp. 477-486. 2008

3. T. Kenesei, B. Feil, J. Abonyi, Visualization and Complexity Reduction of Neural Networks *Applications of soft computing: updating the state of art.*, pp. 43-52. 2009. *Advances in soft computing; ISBN:9783540880783* vol. 52.

4. T. Kenesei, B. Feil, J. Abonyi, Complexity Reduction of Local Linear Models Extracted from Neural Networks, *Acta Agraria Kaposváriensis*, Volume 11 No 2 2007, ISSN 1418-1789s, 259-271

5. T. Kenesei, J. Abonyi, Interpretable Support Vector Machines in Regression and Classification- Application in Process Engineering, *Hungarian Journal of Industrial Chemistry*, Veszprém 2007, VOL 35. 101-108

6. T. Kenesei, J. Abonyi, Interpretable Support Vector Regression, Artificial Intelligence Research, Vol 1 (2), ISSN:1927-6974 ,2012

7. T. Kenesei, J. Abonyi, Hinging hyperplane based regression tree identified by Fuzzy Clustering and its application, *Applied Soft Computing*,ISSN:1568-4946, vol 13(2) pp. 782-792 2013.

Book Chapter

8. Kenesei Tamás, Madár János, Abonyi János: Adatbányászat, a hatékonyság eszköze (Gyakorlati útmutató kezdõknek és haladóknak)Regressziós technikák, Computerbooks, 2006

Refereed Presentations

9. Hinging hyperplane based Regression tree identified by Fuzzy Clustering *WSC16 - 16th Online World Conference on Soft Computing in Industrial Applications* 2011

10. Visualization and Complexity Reduction of Neural Networks *WSC12 -12th Online World Conference on Soft Computing in Industrial Applications*, 2007

11. Fuzzy Clustering for the Identification of Hinging Hyperplanes Based Regression Trees *WILF - International Workshop on Fuzzy Logic and Application*, 2007

12. Identification of Dynamic Systems by Hinging Hyperplane Models *ICAI 2007 - 7th International Conference on Applied Informatics Eger 2007.*

Non-Refereed Presentations

13. T. Kenesei, B. Balasko, J. Abonyi, A MATLAB Toolbox and its Web based Variant for Fuzzy Cluster Analysis, *Magyar Kutatók Nemzetközi Szimpóziuma*, Budapest 2006

14. Tamas Kenesei, Balazs Feil, Janos Abonyi, Identification of Hinging Hyperplane Models by Fuzzy c-Regression Clustering, *Magyar Kutatók Nemzetközi Szimpóziuma*, Budapest 2006

15. Hinging Hyperplane Model based Control of Hammerstein Systems *Mûszaki Kémiai Napok*, Veszprém, 2007.

16. Complexity Reduction of Local Linear Models Extracted from Neural Networks *VI. Alkalmazott Informatika Konferencia* Kaposvár, 2007.

17. Petroleum Supply Chain Optimization with Linear Programming *APS Forum*, Balatonfüred, 2010.

18. Interactive training of neural network models *27th International Workshop on Chemical Engineering Mathematics* Veszprém, 2007.

Other

19. Kenesei Tamás, Neurális hálózatok alkalmazási lehetõségei, *Molekulák biológiai aktivitásának adat-alapú becslésére alkalmas algoritmusok áttekintése*; 2006

20. Kenesei Tamás, Outlierek hatásainak kiszûrésére alkalmas regressziós modellek elõállítása és alkalmazási lehetõségei, *Molekulák biológiai aktivitásának adat-alapú becslésére alkalmas algoritmusok áttekintése*; 2006

Appendix A

Introduction to regression problems

Linear regression takes place in the following form

$$y = \theta_0 + \sum_{j=1}^{n} \theta_j x_j + \varepsilon \tag{A.1}$$

where ε is the regression error. Given x_j variables definition a regression model means estimation of θ_j parameters. As it was mentioned above to identify a linear regression a so called train datasample is need containing N $\{\mathbf{x}_k, \mathbf{y}_k\}$, $k = 1, \ldots, N$, known data pairs. Function $f(\mathbf{x})$ is estimated by defining a connection between dependent and independent variables in a form of model/function:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \ldots & x_{Nn} \end{bmatrix}, \; \mathbf{Y} = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{N1} & \cdots & y_{Nm} \end{bmatrix} \tag{A.2}$$

In multivariate case a hyperplane is fitted for the best approximation. In this case one of the most used methods are the least square based methods. During identification the LS–based method is minimizing the square of distances between function output and the basic dataset. The distance is called residual:

$$\epsilon_i = y_i - \hat{y}_i \tag{A.3}$$

Aim is to find such a θ parameter set to the N data points where ϵ is minimal:

$$RSS(\theta) = \sum_{i=1}^{N} \left(y_i - \theta_0 - \sum_{i=1}^{N} (x_{ij}\theta_j) \right) = \epsilon^T \epsilon = (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \tag{A.4}$$

$y = f(\mathbf{X}, \theta) + \epsilon$ model, (where $\mathbf{X}$ stands for regression matrix) with matrix representation can form the followings:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \mathbf{Y} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1n} \\ \vdots & & \ddots & \vdots \\ 1 & x_{N1} & \dots & y_{Nm} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_N \end{bmatrix} \tag{A.5}$$

$$\frac{\partial(\mathbf{y} - \mathbf{X}\widehat{\theta})^T(\mathbf{y} - \mathbf{X}\widehat{\theta})}{\partial\widehat{\theta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\widehat{\theta} = 0 \tag{A.6}$$

by rewriting the equation, θ can be computed as follows:

$$\widehat{\theta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{A.7}$$

Linearity: Dependent variable is linear combination of the independent variables. $E(y|x_1, x_2, \dots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$.

Independence: Az ε is independent in any observation.

Unbiased: $E(\varepsilon) = 0 \; \forall \, i = 1, \dots, N$.

Homogenity: Az ε standard deviation equals with the unknown value of σ.

Normal disrtibution: ε has normal distribution.

Regression linear in parameters is such a model which is linear to it's parameters. These models can be represented as

$$\widehat{\mathbf{y}} = \theta_0 + \sum_{k=1}^{M} \theta_k f_k(\mathbf{x}) \tag{A.8}$$

where $f_1, \dots, f_M$ are nonlinear functions while $\theta_0, \dots, \theta_M$ are model parameters and $\widehat{\mathbf{y}(k)}$ is the model output and $\mathbf{x}$ is the regressor vector. See Table A.1 for f_i examples

The mentioned LS technique could be used for parameter estimation with minimizing the following cost function:

$$E = \sum_{i=1}^{N} \left(\mathbf{y} - \left(\theta_0 + \sum_{k=1}^{M} \theta_k f_k(\mathbf{x}) \right) \right)^2 \tag{A.9}$$

where N is the number of datapoints. The X regression matrix (see (A.5) for

Function	Basis Function	$\tilde{y} = \tilde{a} + \tilde{b} \cdot \tilde{x},\ \tilde{a}, \tilde{b} \in \theta$			
		$\tilde{y}$	$\tilde{x}$	$\tilde{a}$	$\tilde{b}$
$y = ab^x$	$\log y = \log a + x \log b$	$\log y$	x	$\log a$	$\log b$
$y = a + b \log x$		y	$\log x$	a	b
$y = ax^b$	$\log y = \log a + b \log x$	$\log y$	$\log x$	$\log a$	b
$y = a + b\frac{1}{x}$		y	$\frac{1}{x}$	a	b
$y = \frac{1}{a+bx}$	$\frac{1}{y} = a + bx$	$\frac{1}{y}$	x	a	b
$y = \frac{A}{1+ce^{bx}}$	$\ln\left(\frac{A}{y} - 1\right) = \ln c + bx$	$\ln\left(\frac{A}{y} - 1\right)$	x	$\ln c$	b

Table A.1: Functions with the ability to transform them to linear forms

linear case) is:

$$\mathbf{X} = \begin{bmatrix} 1 & f_1(x_1) & \dots & f_M(x_1) \\ \vdots & & \ddots & \vdots \\ 1 & f_1(x_N) & \dots & f_M(y_N) \end{bmatrix} \tag{A.10}$$

Regression nonlinear in parameters is a general case of nonlinear regression. ([32]):

$$\widehat{y}_i = f\left(x(i), \theta\right) \tag{A.11}$$

where f function is nonlinear to θ parameters. Main advantage of this type of regression is the generality. However, determining regression parameters is a much more slower process, and it can happen that it is infeasible to find proper parameters for a suitable function. A much more difficult situation implied by this type of regression, since nonlinear local or global optimization schemes should be applied for parameter estimation.[1]- If nonlinear optimization is to be applied the gradient of the model output with respect to the parameters is important:

$$E = \frac{1}{2}\sum_{i=1}^{N} e\left(i\right)^2 = \frac{1}{2}\sum_{i=1}^{N} \left(y_i - \widehat{y}_i\right)^2 \tag{A.12}$$

For the application of gradient-based optimization technique the gradient of the loss function with respect to each θ paramater is required:

$$\frac{\partial E}{\partial \theta} = \sum_{i=1}^{N} e\left(i\right)\frac{\partial e\left(i\right)}{\partial \theta} = -\sum_{i=1}^{N} e\left(i\right)\frac{\partial \widehat{y}_i}{\partial \theta} \tag{A.13}$$

Hence, the gradient of the model output with respect the parameters $\frac{\partial \widehat{y}_i}{\partial \theta}$ is required.

Appendix B

n–fold cross validation

n-fold cross-validation

This technique is intended to avoid the possible bias introduced by relying on any one particular division into test and train components, is to partition the original data in several different ways and to compute the average of the performances over the different partitions. When the available data is divided into n part this approach is called *n-fold cross-validation*. Because of the n identification and verification steps, this method is computationally expensive. An extreme variant of this is to split the N training data into a training set of size $N-1$ and test of size 1 and average the squared error on the left-out pattern over the N possible ways of obtaining such partition.

The beauty of LOO for linear in parameter models is that it can be calculated analytically [98]. The fuzzy model is linear in its consequent parameters. Hence, the LOO criteria and its derivatives can be easily used for these models

$$\hat{\sigma}_{LOO}^2 = \frac{\mathbf{y}^T \mathbf{P} \left(\text{diag}(\mathbf{P})\right)^{-2} \mathbf{P}\mathbf{y}}{N} \tag{B.1}$$

where in case of global identification, $\mathbf{P}$ denotes the projection matrix

$$\begin{aligned} \mathbf{y} - \widehat{\mathbf{y}} &= \mathbf{y} - \mathbf{Q}\theta && \text{(B.2)} \\ &= \mathbf{y} - \mathbf{Q}\left(\mathbf{Q}^T\mathbf{Q}\right)^{-1}\mathbf{Q}^T\mathbf{y} && \text{(B.3)} \\ &= \left(\mathbf{I}_N - \mathbf{Q}\left(\mathbf{Q}^T\mathbf{Q}\right)^{-1}\mathbf{Q}^T\right)\mathbf{y} && \text{(B.4)} \\ &= \mathbf{P}\mathbf{y} && \text{(B.5)} \end{aligned}$$

where the $\mathbf{Q}$ matrix contains the N regressors, $\mathbf{y}$ denotes the estimated outputs of the model, and $\mathbf{P}\widehat{\mathbf{y}}$ the vector of the modeling error. The matrix $diag(\mathbf{P})$ is the same size and has the same diagonal as $\mathbf{P}$ but it is zero off-diagonal, and $\mathbf{I}_N$ represents an identify matrix.

Appendix C

Orthogonal least squares

An often applied solution is to prune the identified model trained with classical cost function. In the following, model reduction techniques of this type will be considered. In general it can be stated that linear model reduction methods are preferred to nonlinear ones because they are exhaustively studied and effectively applied for several types of problems. For that purpose the model should be linear in parameters. A possible method family is orthogonal techniques. These methods can roughly be divided into two groups: the rank revealing ones like SVD-QR algorithm and those that evaluate the individual contribution of the rule or local models, like the orthogonal least-squares approach (OLS). This later technique requires more computations, but for system identification purposes it is preferable as it gives a better approximation result. In the remaining part of this paper OLS is applied for rule ranking and model reduction purposes. OLS works as follows (for a throughout discussion see [1]). Consider a general linear in parameters model:

$$\mathbf{y} = \mathbf{Z}\theta + \mathbf{e} \tag{C.1}$$

where $\mathbf{y} = [y_1, \ldots, y_N]^T$ is the measured output, $\mathbf{Z} = [\mathbf{z_1}, \ldots, \mathbf{z_n}]^T$ is the regressor matrix ($\mathbf{z_i} = [z_{i1}, \ldots, z_{iN}]^T$, $i = 1, \ldots, h$ are the regressors) $\theta = [\theta_1, \ldots, \theta_h]$ is the parameter vector and $\mathbf{e} = [e_1, \ldots, e_N]^T$ is the prediction error. OLS transforms the columns of the regressor matrix $\mathbf{Z}$ into a set of orthogonal basis vectors in order to inspect the individual contribution of each regressor. If they were not orthogonal, they could not been inspected individually. An orthogonalization method should be used to perform the orthogonal decomposition $\mathbf{Z} = \mathbf{VR}$ (often the simple Gram-Schmidt method is used), where $\mathbf{V}$ is an orthogonal matrix such that $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{R}$. Substituting $\mathbf{Z} = \mathbf{VR}$ into Eq. C.1, we get $\mathbf{y} = \mathbf{VR}\theta + \mathbf{e} = \mathbf{Vg} + \mathbf{e}$, where

$\mathbf{g} = \mathbf{R}\theta$. Since the columns $\mathbf{v_i}$ of $\mathbf{V}$ are orthogonal, the sum of squares of y_k can be written as

$$\mathbf{y}^T\mathbf{y} = \sum_{i=1}^{h} g_i^2 \mathbf{v}_i^T \mathbf{v}_i + \mathbf{e}^T\mathbf{e} \tag{C.2}$$

The part of the output variance $\mathbf{y}^T\mathbf{y}/N$ explained by regressors is $\sum_{i=1}^{h} g_i^2 \mathbf{v}_i^T \mathbf{v}_i/N$ and an error reduction ratio due to an individual regressor i can be defined as

$$err_i = \frac{g_i^2 \mathbf{v}_i^T \mathbf{v}_i}{\mathbf{y}^T\mathbf{y}},\ i = 1, \ldots, h. \tag{C.3}$$

This ratio offers a simple means of ordering the regressors. As [1] shows, ”there are only two restrictions to the application of this subset selection technique. First, the model has to be linear in parameters. Second, the set of regressors from which the significant ones will be chosen must be precomputed.” This later one is an important restriction because it means that all regressors are fixed during this procedure. By normalized RBF networks and Takagi-Sugeno fuzzy models this requirement is not met, therefore the original version of OLS cannot be applied. It is because the normalization denominator changes with the number of selected rules, thus the fuzzy basis functions (here: regressors) change. To overcome this problem the value of the denominator can be fixed, but in this case interpretability issues are discarded completely. However, OLS can be very useful for various purposes; modified versions of OLS can also be applied to determine the centers of radial basis functions, or to generate Takagi-Sugeno-Kang fuzzy models.

Appendix D

Model of the pH Process

The modeling and control of pH (the concentration of hydrogen ions) in a continuous stirred tank reactor (CSTR) is a well-known control problem that presents difficulties due to the nonlinearity of the process dynamics. The CSTR is shown schematically in Fig. D.1.

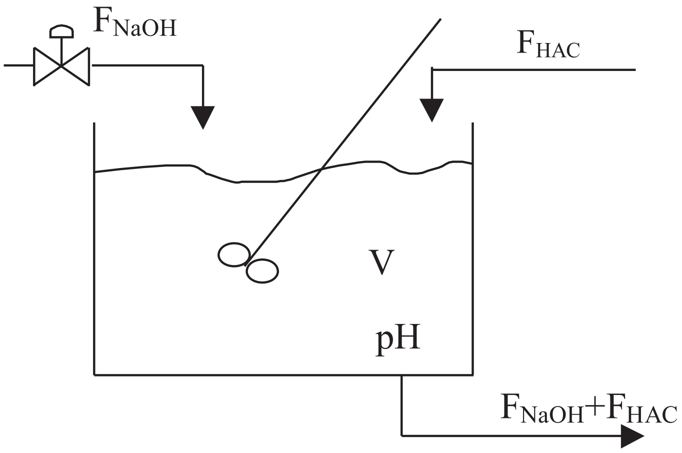

Figure D.1: Scheme of the pH setup.

A dynamic model of the pH in a tank can be obtained by considering the material balances on [Na+] and the total acetate [HAC+AC-] and assuming that acid-base equilibrium and electroneutrality relationships hold [99].

Total acetate balance:

$$F_{HAC}[HAC]_{in} - (F_{HAC} + F_{NaOH})[HAC + AC^{-}] = V\frac{d[HAC + AC^{-}]}{dt}$$

Sodium ion balance:

$$F_{NaOH}[NaOH]_{in} - (F_{HAC} + F_{NaOH})[Na^{+}] = V\frac{d[Na^{+}]}{dt}$$

HAC equilibrium:

$$\frac{[AC^{-}][H^{+}]}{[HAC]} = K_a$$

Water equilibrium:

$$[H^{+}][OH^{-}] = K_w$$

Electroneutrality:

$$[Na^{+}] + [H^{+}] = [OH^{-}] + [AC^{-}]$$

The pH can be calculated from the previous equations as

$$[H^{+}]^3+[H^{+}]^2(K_a+[Na^{+}])+[H^{+}]([Na^{+}]K_a-[HAC+AC^{-}]K_a-K_w)-K_wK_a = 0$$

$$pH = -\log[H^{+}]$$

The parameters used in our simulations are taken from [100] and are given in Table D.1.

Table D.1: Parameters used in the simulations.

Parameter	Description	Nominal Value
V	Volume of the tank	1000 [l]
F_{HAC}	Flow rate of acetic acid	81 [l/min]
F_{NaOH}	Flow rate of NaOH	515 [l/min]
$[NaOH]_{in}$	Inlet conc. of NaOH	0.05 [mol/l]
$[HAC]_{in}$	Inlet conc. of acetic acid	0.32 [mol/l]
$[Na^{+}]$	Initial conc. of sodium in the CSTR	0.0432 [mol/l]
$[HAC + AC^{-}]$	Initial conc. of acetate in the CSTR	0.0432 [mol/l]
K_a	Acid equilibrium constant	1.75310^{-5}
K_w	Water equilibrium constant	10^{-14}

Appendix E

Model of electrical water heater

The schematic diagram of the water-heater is shown in Fig. E.1.

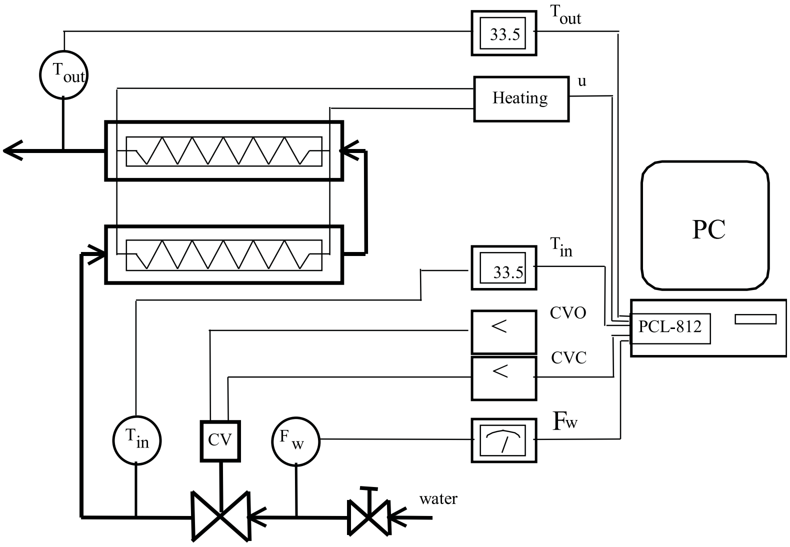

Figure E.1: The scheme of the physical system.

The water comes from the water pipeline into the heater through a control valve and a pair of metal pipes containing a cartridge heater. The control task is to control the T_{out} outlet temperature by adjusting the u heating signal of the cartridge heater.

The temperature measurement is realized by Pt100 thermometers. The system has four analogue inputs (T_{in} inlet temperature, T_{out} outlet temperature, valve position and the F flow-rate), and two digital (open and close of the valve, CVO and CVC) and one analogue output (heating control signal, u). The heaters are linked parallel and have a performance of 1 kW. The process is connected to a PC computer through ADVANTECH LabCard PCLD-780 and PCL-812 data acquisition boards. GENIE 3.02 data acquisition and control software was used to filter and convert the input signals (0-5V). The control algorithm runs in MATLAB 4.2. The program gets the filtered and converted measured data through DDE in every 2 seconds [101].

For the purpose of physical modeling the system was decomposed into four interacting elements: the cartridge-heater (subscript h), the streaming water (subscript w), the pipe wall (subscript p) and the environment (subscript e). The following three heat balances in the form of partial differential equations can be established:

$$\begin{aligned}
V_h \rho_h C_{p_h} \frac{\partial T_h}{\partial t}(t,z) &= Q(u) - \alpha_1 A_1 (T_h - T_w) \\
V_w \rho_w C_{p_w} \frac{\partial T_w}{\partial t}(t,z) + (F\rho C_p)_w \frac{\partial T_w}{\partial z}(t,z) &= \alpha_1 A_1 (T_h - T_w) - \alpha_2 A_2 (T_w - T_p) \\
V_p \rho_p C_{p_p} \frac{\partial T_p}{\partial t}(t,z) &= \alpha_2 A_2 (T_w - T_p) - \alpha_e A_e (T_p - T_e)
\end{aligned}$$

where, $z \in [0, L]$ with L denotes the length of the pipe. The description and the nominal values of the parameters are given in Table E.1.

Table E.1: Parameters used in the simulation model of the heating system.

Parameter	Description	Nominal value
L	Length of the pipe	$2 \times 48010^{-3}\, m$
ϱ_h	Density of the cartridge	$3650\, kg/m^3$
C_{ph}	Heat capacity of he cartridge	$1047\, J/kgK$
A_h	Surface of the cartridge	$2.41 \times 10^{-2}\, m^2$
V_h	Volume of the cartridge	$4.82 \times 10^{-5}\, m^3$
α_1	$h - w$ heat transfer coefficient	$316.3\, Wm^{-2}K^-1$
ϱ_w	Density of the water	$1000\, kg/m^3$
C_{pw}	Heat capacity of the water	$4186\, J/kgK$
T_{in}	Inlet water temperature	$11.8\, C$
V_w	Volume of the water	$1.16 \times 10^{-4}\, m^3$
α_2	$w - p$ heat transfer coefficient	$1196.1\, Wm^{-2}K^-1$
ϱ_p	Density of the wall	$7850\, kg/m^3$
C_{pp}	Heat capacity of the wall	$502\, J/kgK$
t_e	Temperature of the environment	$21.6\, C$
A_p	Inner surface of the wall	$4.46 \times 10^{-2}\, m^2$
V_p	Volume of the wall	$7.37 \times 10^{-5}\, m^3$
A_e	Outer surface of the wall	$5.36 \times 10^{-2}\, m^2$
α_e	$p - e$ heat transfer coefficient	$1015.9\, Wm^{-2}K^-1$

The performance of the cartridge heater is given by:

$$Q(u) = Q_M \left[u - \frac{\sin(2\pi u)}{2\pi} \right] \tag{E.1}$$

where Q_M is the maximal power, and u is the heating signal (voltage). The partial differential equations are approximated by eight compartments of equal volume. As Eq. E.1 shows, the heating performance is a static nonlinear function of the heating signal (control input).

Bibliography

[1] O. Nelles. *Nonlinear system identification*. Springer-Verlag, 2001.

[2] E.H. Mamdani, T. Teraqno, K. Asai, and M. Sugeno. Fuzzy–systems theory and its applications. *Nature*, 359:788–788, 1992.

[3] J.M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proceedings of the IEEE*, 83:345–377, 1995.

[4] N.V. Bhat and T.J. McAvoy. Use of neural nets for dynamic modelling and control of chemical process systems. *Computers and Chemical Engineering*, 4/5:573–585, 1990.

[5] N.V. Bhat, P.A. Minderman, T.J. McAvoy, and N.S. Wang. Modelling chemical process systems via neural computation. *IEEE Control Systems Magazine*, April:24–30, 1990.

[6] E. Hernandez and Y. Arkun. Control of nonlinear systems using polynomial ARMA models. *AICHE Journal*, 39(3):446–460, 1993.

[7] B.E. Ydstie. Forecasting and control using adaptive connectionist networks. *Computers and Chemical Engineering Journals*, 15(4/5):583–599, 1990.

[8] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B Deylon, P-Y Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31:1691–1724, 1995.

[9] J. Abonyi and B. Feil. Computational in-telligence in data mining. *Informatica*, 29:3–12, 2005.

[10] Gregory Piatetsky-Shapiro. *Knowledge discovery in databases*. AAAI Pres, 1991.

[11] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions and Systems, Man, and Cybernetics*, 3:28–44, 1973.

[12] M. Sugeno and G.T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28:15–33, 1988.

[13] M. Sugeno and K. Tanaka. Successive identification of a fuzzy model and its application to prediction of a complex system. *Fuzzy Sets and Systems*, 42:315–334, 1991.

[14] M. Sugeno and T. Yasukawa. A fuzzy–logic–based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31, 1993.

[15] J.-S.R. Jang. Input selection for ANFIS learning. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, volume 2, pages 1493–1499, New York, USA, 1996.

[16] Lofti A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 36(3):77–84, 1994.

[17] Vojislav Kecman. *Learning and Soft Computing*. 2001.

[18] S. Niu and P. Pucar. Hinging hyperplanes for non-linear identification. *http://www.control.isy.liu.se*, 1995.

[19] S. Ernst. Hinging hyperplane trees for approximation and identification. *Decision and Control Proceedings of the 37th IEEE Conference*, 2:1266–1271, 1998.

[20] L. Breiman. Hinging hyperplanes for regression, classification and function approximation. *IEEE Transactions on Information Theory*, 39:311–325, 1993.

[21] . Nunez, C. Angulo, and A. Catala. Rule extraction from support vector machines. *European Symposium on Artificial Neural Networks Proceedings*, pages 107–112, 2002.

[22] Y.X. Chen and J.Z. Wang. Support vector learning for fuzzy rule-based classification systems. *IEEE Trans. Fuzzy Systems*, 11:716–728, 2003.

[23] Y.X. Chen and J.Z. Wang. Kernel machines and additive fuzzy systems: classification and function approximation. *Proceedings of IEEE International Conference on Fuzzy Systems*, pages 789–795, 2003.

[24] Corrina Cortes and Vladimir Vapnik. Support-vector networks. *AT&T Research Labs, USA*, 1995.

[25] QJacek M. Leski. On support vector regression machines with linguistic interpretation of the kernel matrix. *Fuzzy Sets and Systems*, 157:1092–1113, 2006.

[26] Y.L. Cun, J. Denker, and S. Solla. Optimal brain damage. *Advances in neural information processing systems*, 2:598–605, 1990.

[27] W. Duch. Coloring black boxes: visualization of neural network decisions. *Int. Joint Conf. on Neural Networks Portland*, 1:1735–1740, 2003.

[28] W. Duch. Visualization of hidden node activity in neural networks: I. visualization methods. *Lecture Notes in Artificial Intelligence*, 3070:38–43, 2004.

[29] W. Duch. Visualization of hidden node activity in neural networks: Ii. application to rbf networks. *Lecture Notes in Artificial Intelligence*, 3070:44–49, 2004.

[30] H.M. Henrique, E.L. Lima, and D.E. Seborg. Model structure determination in neural network models. *Chemical Engineering Science*, 55:5457–5469, 2000.

[31] J.M. Benitez, J.L. Castro, and I. Requena. Are artifical neural networks black boxes? *IEEE Transactions on Neural Networks*, 8(5):1156–1164, 1995.

[32] J. Abonyi. *Fuzzy model identification for control.* Birkhäuser Boston, 2003.

[33] D.R. Ramirez, E.F. Camacho, and M.R. Arahal. Implementation of min-max mpc using hinging hyperplanes to a heat exchanger. *Control Engineering Practice*, 12:1197–1205, 2004.

[34] P. Pucar and J. Sjoberg. Parameterization and conditioning of hinging hyperplane models. *Technical Report LiTH-ISY-R-1809 epartment of Electrical Engineering, Linkoping University*, 40:37–50, 1995.

[35] Chengtao Wen, Shuning Wang, Xuexiang Jin, and Xiaoyan Ma. Identification of dynamic systems using piecewise-affine basis function models. *Automatica*, 43:1824–1831, 2007.

[36] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40:37–50, 2004.

[37] R. Hathaway and J. Bezdek. Switching regression models and fuzzy clustering. *IEEE Transactions on fuzzy systems*, 3:195–204, 1993.

[38] J. Abonyi and B. Feil. *Cluster Analysis for Data Mining and System Identification.* Birkhäuser Basel, 2007.

[39] J. Abonyi T. Kenesei. Hinging hyperplane based regression tree identified by fuzzy clustering. *WSC16 - 16th Online World Conference on Soft Computing in Industrial Applications*, 2011.

[40] Robert Babuska Antonio Sala, Thierry Marie Guerrab. Perspectives of fuzzy systems and control. *Fuzzy Sets and Systems*, 156:432–444, 2005.

[41] Hans Hellendoorn Radu Precup. A survey on industrial applications of fuzzy control. *Computers in Industry*, 63:213–226, 2011.

[42] Gang Feng. A survey on analysis and design of model-based fuzzy control systems. *IEEE Transactions on Fuzzy Systems*, 14(5):676–697, 2011.

[43] R. Babuška. *Fuzzy Modeling for Control.* Kluwer Academic Publishers, Boston, 1998.

[44] J. Abonyi, R. Babuska, M. Setnes, H.B. Verbruggen, and F. Szeifert. Constrained parameter estimation in fuzzy modeling. In *Proceedings FUZZ-IEEE'99*, pages 951–956, 1999.

[45] J. Abonyi, R. Babuska, , F. Szeifert, and L. Nagy. Identification and control of nonlinear systems using fuzzy hammerstein models. *Industrial and Engineering Chemistry Research*, 39(11):4302–4314, 2000.

[46] M. Kubat. Decision trees can initialize radial-basis-function networks. *IEEE Trans. NN*, 9:813–821, 1998.

[47] Chengtao Wen and Xiaoyan Ma. A max-piecewise-linear neural network for function approximation. *Neurocomputing*, 71:843–852, 2008.

[48] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[49] P. Pucar and J. Sjoberg. On the hinge finding algorithm for hinging hyperplanes. *IEEE Transaction on Information Theory*, 44:1310–13109, 1998.

[50] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50:1567–1280, 2005.

[51] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.

[52] Qianfeng Cai, Zhifeng Hao, and Xiaowei Yang. Gaussian kernel-based fuzzy inference systems for high dimensional regression. *Neurocomputing - Article in press*, 2011.

[53] K. Weierstrass. Über continuirliche functionen eines reellen arguments die für keinen werth des letzeren einen bestimmten differentialquotienten besitzen. *Königliche Akademie der Wissenschaften*, 1872.

[54] K. Weierstrass. Über die analytische darstellberkeit sigenannter willkürlicher functionen einer reellen veranderlichen. *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften zu Berlin*, 1885.

[55] D. Hilbert. Mathematische probleme. *2. International Congress of Mathematicians, Paris, France*, 1900.

[56] V.I. Arnold. On functions of three variables. *Dolkady Akademii Nauk USSR*, 114:679–681, 1957.

[57] A.N. Kolmogorov. On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition. *Dolkady Akademii Nauk USSR*, 114:953–956, 1957.

[58] D.A. Sprecher. On the structure of continuous functions of several variables. *Trans. American. Math. Soc.*, 115:340–355, 1965.

[59] G.G. Lorentz. *Approximation of Functions*. 1966.

[60] J.P. De Figueiredo. Implications and applications of kolmogorov's superposition theorem. *IEEE Tr. Atoumated Control*, 25(6):1127–1231, 1980.

[61] M. Stinchcombe & H. White K. Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[62] E.K. Blum & L.K. Li. Approximation theory and feedforward networks. *Neural Networks*, 4(4):511–515, 1991.

[63] V. Kurková. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5:501–506, 1992.

[64] L.Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[65] L.X. Wang. Fuzzy systems are universal approximators. *IEEE Int. Conf. on Fuzzy Systems, San Diego, USA*, pages 1163–1169, 1992.

[66] B. Kosko. Fuzzy systems are universal approximators. *IEEE Trans. on Computers.*, 43(11):1329–1333, 1994.

[67] J.L. Castro. Fuzzy logic controllers are universal approximators. *IEEE Trans. on SMC.*, 25:629–635, 1995.

[68] B. Moser. Sugeno controllers with a bounded number of rules are nowher dense. *Fuzzy Sets and Systems*, 104(2):269–277, 1999.

[69] D. Tikk. On nowhere denseness of certain fuzzy controllers containing prerestricted number of rules. *Tatra Mountains Math. Publ.*, 16:369–377, 1999.

[70] L.T. Kóczy & B. Moser E.P. Klemet. Are fuzzy systems universal approximators? *Int. J. General Systems*, 28(2):259–282, 1999.

[71] M.H. Stone. Applications of the theory of boolean rings to general topology. *Transactions of the American Mathematical Society*, 41(3):375–481, 1937.

[72] M.H. Stone. The generalized weierstrass approximation theorem. *Mathematics Magazin*, 21(4):167–184, 1948.

[73] N.E. Cotter. The stone–weierstrass theorem and its application to neural networks. *IEEE Transaction on Neural Networks*, 1(4):290–295, 1990.

[74] L.X. Wang & J.M. Mendel. Fuzzy basis functions, universal approximators and orthogonal least squares learning. *IEEE Transaction on Neural Networks*, 3:807–814, 1992.

[75] R. Setiono, W.K. Leow, and J.Y.L. Thong. Opening the neural network black box: an algorithm for extracting rules from function approximating artificial neural networks. *Proceedings of the 21st International Conference on Information systems–Queensland*, pages 176–186, 2000.

[76] L.N. de Castro & E.M. Iyoda & F.J.V. Zuben & R. Gudwin. Feedforward neural network initialization: an evolutionary approach. *Proceedings of the 5th Brazilian Symposium on Neural Networks*, pages 43–49, 1998.

[77] U. Markowska-Kaczmar & M. Chumieja. Opening neural network black box by evolutionary approach. *Design and application of hybrid intelligent systems*, pages 147–156, 2003.

[78] N.V. Bath and T.J. McAvoy. Determining model structure for neural models by network stripping. *Computers chem. Engng*, 16(4):271–281, 1992.

[79] B. Hassibi, D. Stork, and G. Wolff. Optimal brain surgeon and general network pruning. *Technical report*, 1992.

[80] Marcell Olajos & Tibor Chovan & Stefan Mittermayr & Tamas Kenesei & Peter Hajos & Imre Molnar & Ferenc Darvas & Andras Guttman. Artificial neural network modeling of ph dependent structural descriptor-mobility relationship for capillary zone electrophoresis of tripeptides. *Journal of Liquid Chromatography & Related Technologies*, 31:2348–2362, 2008.

[81] J.H. Chiang and P.Y. Hao. Support vector learning mechanism for fuzzy rule-based modeling: a new approach. *IEEE trans. Fuzzy Syst.*, 12:1–12, 2004.

[82] Q. Cai & Z. Haom & X. Yang. Gaussian kernel-based fuzzy inference system for high dimensional regression. *Neurocomputing*, 2011.

[83] . L. Castro, L. D. Flores-Hidalgo, C. J. Mantas, and J. M. Puche. Extraction of fuzzy rules from support vector machines. *Fuzzy Sets and Systems*, 158(18):2057–2077, 2007.

[84] Dug Hun Hong and Changha Hwang. Support vector fuzzy regression machines. *v*, 138(2):271–281, 2003.

[85] Steve R. Gunn. Support vector machines for classification and regression. *Technical report*, 1998.

[86] Chia-Feng Juang and Cheng-Da Hsieh. Ts-fuzzy system-based support vector regression. *Fuzzy Sets and Systems*, 160(17):2486–2504, 2009.

[87] Asli Celikyilmaz and I. Burhan Türksen. Fuzzy functions with support vector machines. *Information Sciences*, 177(23):5163–5177, 2007.

[88] Hsiao-Fan Wang and Ruey-Chyn Tsaur. Insight of a fuzzy regression model. *Fuzzy Sets and Systems*, 112(3):355–369, 2000.

[89] L.M.C. Buydens B. Üstün, W.J. Melssen. Visualisation and interpretation of support vector regression models. *Analytica Chimica Acta*, 595(1–2):299–309, 2007.

[90] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Ratsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Trans. on Neural Networks*, 10(5), 1999.

[91] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke. Similarity measures in fuzzy rule base simplification. *IEEE Trans. SMC-B*, 28:376–386, 1998.

[92] J. Yen and L. Wang. Simplifying fuzzy rule-based models using orthogonal transformation methods. *IEEE Trans. SMC-B*, 29:13–24, 1999.

[93] M. Setnes and H. Hellendoorn. Orthogonal transforms for ordering and reduction of fuzzy rules. *FUZZ-IEEEE San Antonio Texas*, page 700.705, 2000.

[94] A. M. Smola & B. Schölkopf. A tutorial on support vector regression. *NeuroCOLT Tehnical Report*, 2003.

[95] B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansion, and an interpretation of clustering as approximation in feature spaces. *DAGM-Symposium, Informatik aktuell*, pages 124–132, 1998.

[96] Y. Jin. Fuzzy modeling of high-dimensional systems. *IEEE Trans. FS*, 8:212–221, 2000.

[97] M. Setnes and R. Babuška. Rule base reduction: some comments on the use of orthogonal transforms.

[98] M.J.L. Orr. Introduction to radial basis function networks. In *Research Report*. Centre of Cognitive Science, University of Edinburgh, Edinburgh, Schotland, April 1996.

[99] T.J. McAvoy, E. Hsu, and S. Lowenthal. Dynamics of pH in controlled stirred tank reactor. *Industrial Engineering Chemical Process Design and Development*, 11(1):68–70, 1972.

[100] N.V. Bhat and T.J. McAvoy. Determining model structure for neural models by network stripping. *Computers and Chemical Engineering*, 16:271–281, 1992.

[101] J. Abonyi, A. Bodizs, L. Nagy, and F. Szeifert. Hybrid fuzzy convolution model based predictor corrector controller. In M. Mohamadian, editor, *Computational Intelligence for Modelling Control and Automation*, pages 265–270. IOS Press, Holland, ISBN 9–051–99474–5 1999.