VTT INFORMATION TECHNOLOGY

RESEARCH REPORT TTE1-2000-43

FUME Project

Discovery of Fuzzy Models from Observation Data

Version 1.0

December 2000

Mikko Hiirsalmi
Evangelos Kotsakis
Antti Pesonen
Antoni Wolski

Version history

Version	Date	Author(s)	Reviewer	Description
0.1-1	15.12.2000	all		First draft
1.0	28.12.2000	all		First version

Contact information

Antoni Wolski
VTT Information Technology
P.O. Box 1201, FIN-02044 VTT, Finland
Street Address: Tekniikantie 4 B, Espoo
Tel. +358 9 4561, fax +358 9 456 6027
Email: antoni.wolski@vtt.fi
Web: http://www.vtt.fi/tte/projects/fume/

Last modified on 18 January, 2001
T:\Fume\Reports\discovery-report\main-newest.doc

Abstract

Methods for automatic identification of fuzzy models for the purposes of real-time industrial process monitoring are studied and tested. Theoretical fuzzy and neurofuzzy approaches to identification of Mamdani and Takagi-Sugeno inference models are summarized. Support of fuzzy inference in the active database system RapidBase is discussed. Commercial products Matlab and fuzzyTech are tested in a case study of predicting abnormal states in a waste water treatment plant. The most challenging part of the modeling turns out to be the structural identification, that is the derivation of the rule format and the selected variables. Best results are produced, in Matlab, by using neurofuzzy learning (ANFIS) together with the sequential forward search and, in fuzzyTech, with the learning of degree of rule support. Applicability of the studied methods to automatic extraction of RapidBase fuzzy monitoring models from measurement data is discussed.

Table of Contents

1 Introduction

After Mamdani [Man74] first introduced a fuzzy logic based controller, giving life to fuzzy inference, the technology has been spreading ever since. The reasoning model based on Zadeh's fuzzy IF THEN rules found its way into various automatic control applications. The Mamdani's model yielded well to traditional knowledge acquisition from human specialists. The rules were intuitive and understandable in common terms. Later, Takagi and Sugeno [ST83] introduced another model that was better suited for automatic model construction. In recent years, the area of fuzzy inference systems (FIS) and the methodologies of FIS model construction (called also model identification or learning) has been subject of intensive research. In addition to control systems, FIS solutions have been applied to other tasks requiring computational intelligence, like real-time monitoring of industrial processes. For this purpose, we have developed an active database system called RapidBase [WKLP00] equipped with fuzzy triggers [WB98]. In our efforts to apply RapidBase to real-life problems, we noticed that FIS models of large processes become large and incomprehensible. The need for efficient and reliable model construction methods has become more and more acute.

In this report, we concentrate on methods for extraction of multivariable FIS models from observation data, for the purposes of real-time process monitoring. The task of the FIS model is to detect, in real-time, abnormal process states, especially such that may be projected to a malfunction in a future, and require a corrective action on the part of a human operator. A case study is also described where a model predicting states of waste water treatment process is being constructed with commercial fuzzy tools.

We survey the basic FIS models, the RapidBase implementation, and the limitations of the model identification in Section 2. Section 3 is devoted to theoretical methods of model identification using both fuzzy and neurofuzzy approaches. The case study is introduced in Section 4, and various experiments are described. Applicability to RapidBase is also discussed. The results are summarized in Section 5.

2 Fuzzy concepts and modeling methods

In this Section we address the basic fuzzy modeling methods starting with the basic concepts of fuzzy logic, explaining the fuzzy inference methods, introducing the fuzzy methods implemented in RapidBase and finally, addressing what can be learned from example data.

2.1 Basic concepts

The concept of fuzzy logic was introduced by Lotfi Zadeh in early 60's [Zad65]. The fuzzy logic is based on the *fuzzy set theory* and especially on the concept of a *fuzzy set*.

Informally, a fuzzy set is a set with imprecise boundaries in which the transition from membership to non-membership is gradual rather than abrupt. A fuzzy set F in a universe of discourse U is characterized by a *membership function* μ_F, which associates each element $u \in U$ with a grade of membership $\mu_F(u) \in [0, 1]$ in the fuzzy set F. Note that a classical set A in U is a special case of a fuzzy set with all membership values $\mu_A(u) \in \{0, 1\}$.

A *fuzzy type T* is a mapping of related membership functions (also called *fuzzy terms*) against a specific universe of discourse U. Let us have a look at an example of a fuzzy type to clarify the idea.

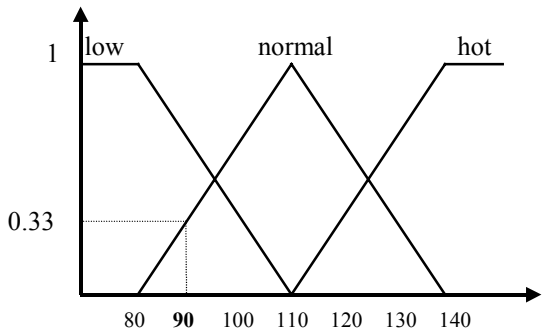


Figure 2-1. The fuzzy type Temperature.

The fuzzy type Temperature is defined in the figure above. Its term set *T(Temperature)* is *{low, normal, hot}*. We interpret "low" as "a temperature below about 110°C," "normal" as "a temperature close to 110°C," and "hot" as "a temperature above about 110°C". These *linguistic terms* can be characterized as fuzzy sets whose membership functions are shown in the Figure. Each element $u \in U$ belongs to each fuzzy set defined with a degree of membership $\mu_F(u) \in [0, 1]$. For example, if the temperature is 90°C then the membership degree for the fuzzy subset *low* is equal to 0.33. At the same point the membership degree for the fuzzy subset *hot* equals 0.

A *fuzzy proposition* is of the form X IS A, where X is a *linguistic variable* of a fuzzy type T and A is a fuzzy set (linguistic term) defined on T. A *fuzzy rule* takes the form of an *if-then* statement such as "if X IS A then Y IS B" where X IS A and Y IS B are fuzzy propositions. The *if* part of a fuzzy *if-then* rule is called the *antecedent* (or premise), whereas the *then* part is called the *consequent*. The antecedent part of a fuzzy rule is a conjunction and/or a disjunction of fuzzy propositions.

A *fuzzy implication* is viewed as describing a fuzzy relation between the fuzzy sets forming the implication [MJ94]. A fuzzy rule, such as "if X IS A then Y IS B" is implemented by a fuzzy implication (fuzzy relation) which has a membership function $\mu_{A \to B}(x, y) \in [0, 1]$. Note that $\mu_{A \to B}(x, y)$ measures the degree of truth of the implication relation between x and y. In control applications, *Mamdani implication* (minimum) is one of the most commonly used interpretation for the implication, it is defined as:

$$\mu_{A \to B}(x, y) = \min[\mu_A(x), \mu_B(y)]$$

A set of related fuzzy rules forms a *fuzzy rule base* that can be used to infer *fuzzy results* in the form of fuzzy sets. A fuzzy result can be further refined to a more useful crisp

result in the process called *defuzzification*. The most common mean of defuzzification is called the *center of gravity* method in which the center of gravity of the fuzzy set is measured and projected to the x-axis to get the crisp result. For an example, see the figure below.

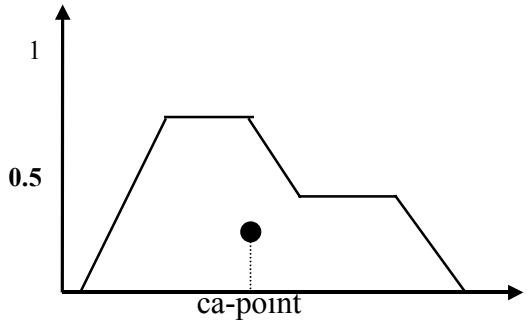


Figure 2-2. Center of gravity (ca) defuzzification.

2.2 Inference using fuzzy logic

The most commonly used fuzzy inference method is *the Max-Min inference method* or *Mamdani inference method*. Another popular fuzzy model structure is called the *Takagi-Sugeno model* [TS85]. An overview of both the methods is given in this Section. For a more detailed description of the models, see Section 3.1 *Fuzzy model extraction methods*.

2.2.1 Mamdani model

Let us consider the following rule base (where X, Y and Z are linguistic variables).

R_i: *if X is* A_i *and Y is* B_i *then Z is* C_i $i = 1..n$

Given the input fact (x_0, y_0), the goal is to determine the output "Z is C". The first step to make is to fuzzify the given input. The fuzzifier maps the input data $x_0 \in U_x$ into the fuzzy set A* and $y_0 \in U_y$ into the fuzzy set B*.

The next step is to evaluate the truth value for the premise of each rule, and then apply the result to the conclusion part of each rule using the fuzzy implication. The membership functions defined on the input variables are applied to their actual values to determine the degree of truth for each rule premise. The degree of truth for a rule's premise is computed in our example rule base as follows:

$$\alpha_i = \mu_{Ai \text{ and } Bi}(x_0, y_0) = \min(\mu_{Ai}(x_0), \mu_{Bi}(y_0))$$

If a rule's premise has nonzero degree of truth then the rule is activated. The next step is to find the output, C'_i, of each of the rules:

$$\mu_{C'i}(w) = \mu_{(Ai \text{ and } Bi) \rightarrow Ci}(x_0, y_0, w), \forall w \in W$$

In Min inferencing (or Mamdani implication rule) the implication is interpreted as a fuzzy *And* operator:

$$\mu_{C'i}(w) = \mu_{(Ai \text{ and } Bi) \rightarrow Ci}(x_0, y_0, w), \forall w \in W$$

$= \mu_{Ai \text{ and } Bi}(x_0, y_0)$ and $\mu_{Ci}(w) = \min(\mu_{Ai \text{ and } Bi}(x_0, y_0), \mu_{Ci}(w))$

In the rule aggregation step, all fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. The purpose is to aggregate all individual rule outputs to obtain the overall system output. In the Max composition, the combined output fuzzy subset C* is constructed by taking the maximum over all of the fuzzy subsets assigned to the output variable by the inference rule:

$$\mu_{C*}(w) = \max(\mu_{C'1}(w), \mu_{C'2}(w), \ldots, \mu_{C'n}(w))$$

Normally, the defuzzification step is executed as the last step and the most commonly used method is the center of gravity described earlier.

2.2.2 Takagi-Sugeno model

The Takagi-Sugeno fuzzy model differs from the Mamdani model by introducing crisp functions as the consequences of the rules. This structure offers a systematic approach to generate fuzzy rules from a given input-output data set. A Takagi-Sugeno rule set is of the form:

R_i: *if X is* A_i *and Y is* B_i *then* $z_i = f_i(x_0, y_0)$, $i = 1..n$, (x_0, y_0) *is the input*

The antecedent of each rule is a set of fuzzy propositions connected with the AND operator. The consequent of each rule is a crisp function of the input vector $[x_0, y_0]$. By means of the fuzzy sets of the antecedent propositions the input domain is softly partitioned in smaller regions where the mapping is locally approximated by the crisp functions f_i.

Combining the rules and their effects differ from the Mamdani method considerably. One variation of the Takagi-Sugeno inference system uses the weighted mean criterion to combine all the local representations in a global approximator, like this:

$$z = \frac{\sum_{i=1}^{r} \mu_i z_i}{\sum \mu_i}$$

where μ_i is the degree of fulfillment of the *i*th rule and *r* is the number of rules in the rule base.

2.3 RapidBase fuzzy methods

The fuzzy inference engine in RapidBase is tightly connected to the trigger system of the RapidBase Server. A fuzzy trigger, when fired, starts the fuzzy reasoning process that is executed by the inference engine. RapidBase implements the Mamdani method to infer the fuzzy results. For the membership function type the triangular and trapezoidal function types can be used. The number of fuzzy propositions in an antecedent of a rule is not limited. Only one proposition is accepted as a consequent of a rule. Center of gravity is used to defuzzify the fuzzy result when a crisp result is needed.

In addition to basic Mamdani method, the following features are added: Degree of Support (DoS) of each rule in a rule set, AND and OR operators between antecedent fuzzy

propositions and negation of a fuzzy proposition. An example of a RapidBase fuzzy rule follows.

(0.8) IF motor IS hot AND torque IS NOT high THEN alarm IS high

Also, fuzzy quantifiers and finally fuzzy temporal restrictors, addressed in sequel, are implemented in RapidBase.

2.3.1 Fuzzy quantifiers

Classical logical systems use two quantifiers: *universal* and *existential*. Fuzzy logic admits a wide variety of fuzzy quantifiers exemplified by *few*, *several*, *about ten*, etc. A quantified fuzzy proposition is of the form Q X's ARE A, where Q is a fuzzy quantifier, X is a set of objects and A is a fuzzy set. A fuzzy quantifier is defined with a membership function. For example, the following membership function could be defined for the fuzzy quantifier *most*.

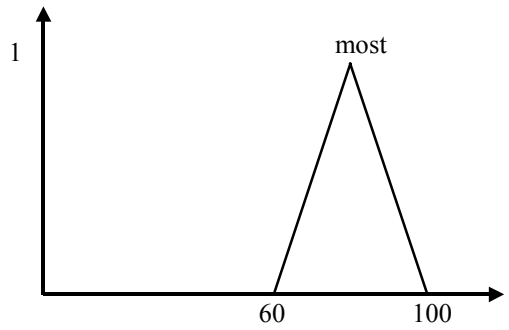


Figure 2-3. The fuzzy quantifier 'most'.

For the calculus used to evaluate quantified propositions in RapidBase, please refer to [PW00].

2.3.2 Fuzzy temporal restrictors

A fuzzy temporal restrictor is an entity that restricts a fuzzy proposition temporally. A temporally restricted fuzzy proposition can be written as “A *HAS BEEN* B T” which means A is satisfying the fuzzy predicate B, taking into account the temporal restrictor T.

A temporal restrictor is defined with a membership function. The following membership function defines the temporal restrictor *few_minutes_ago*.

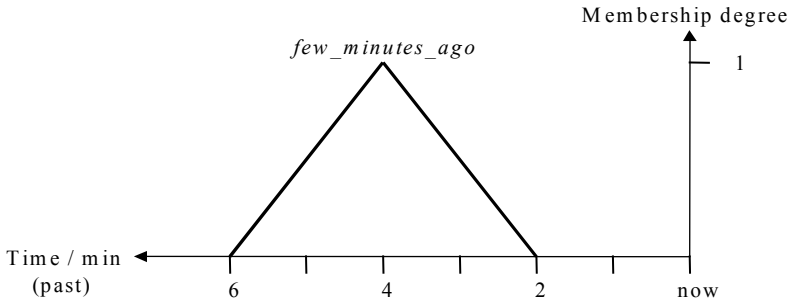


Figure 2-4. The membership function of a temporal restrictor.

The origin in the definition describes the dynamic concept *now* that means the current instance of time. The membership function *few_minutes_ago* is defined to have a positive membership degree between “six minutes ago” and “two minutes ago”. Current time is changing all the time, so, in fact, the membership function definition is also constantly changing. Using the temporal restrictor we can define, for example, the following proposition:

IF motors WERE HOT few_minutes_ago THEN "issue medium alarm"

For the calculus used to evaluate temporally restricted propositions in RapidBase, please refer to [PW00].

2.4 Using example data to build fuzzy inference models

Fuzzy models should be built using expert prior knowledge when available. Learning from example data may be used for tuning an existing fuzzy inference system and also for automated model extraction from the data. Bonissone [BCGK99] provides a good survey of building fuzzy inference models combining different methods from classical control theory and soft computing and combining expert knowledge with data based tuning. Also Babuska [BVH99] describes interesting methods for fuzzy modeling also involving data based methods.

Learning from data presumes the existence of measurement data records that represent the modeled phenomena adequately. There has to be enough data describing the system while it is at a stable state. The measurements should describe the same phenomena.

There are different types of learning schemes available depending on the type of data that is available. In supervised learning, there exists a clear output (possibly provided by a reliable teacher) for each input data vector and the task is to learn an adequate input-output mapping from the data. Reinforcement learning may be applied in cases where the feedback is given occasionally, may be delayed and is only partially targeted. Unsupervised learning is used when there is no feedback available and the aim is to find internal structure within the input space. Typically clustering is used to form clusters of measurements where the similarity between vectors within a cluster is as close as possible and vectors from different clusters are as dissimilar as possible.

Typically the current automated learning algorithms require that all the knowledge from the system has been represented as input feature vectors with the same amount of features

(all data needs to be on the same vector format). Possible missing values need to be pre-processed appropriately and the studied phenomena has to be represented in the vector format by calculating appropriate features describing the raw data.

Automated learning algorithms exist supporting many different types of model classes: neural networks (eg. MLP, RBF, SOM), decision trees, association rules, episode rules, logic programs and Bayesian belief networks. Fuzzy inference systems (FIS) may be viewed as a particular model class and also some of the more traditional model types may be fuzzified (fuzzy decision trees, fuzzy clustering and fuzzy SOM). Learning may be divided into structural learning (system identification in classical system theory) and parameter learning (parameter estimation in classical system theory). Structure determines the flexibility of the model in the approximation of the mappings. There is a trade-off with good generalization ability and model complexity. More complex models may be fitted well to the training data but they tend to perform much worse on unseen validation data indicating low generalization capability. One should aim at building models with a suitable complexity.

In fuzzy model extraction, structural learning may involve the selection of the most informative inputs to use (called variable selection) and selection of the order of the system (number of input and output lags). Also the selection of the model type may be involved (Takagi-Sugeno or Mamdani inference model), as well as the identification of the number of rules to use in the mapping. This has to do with fixing the granularity of the fuzzy partition of the input space (selecting the number and type of membership functions).

In fuzzy model extraction, parameter learning involves the tuning of the membership function parameters for the inputs and the consequents. Also the other structural parameters could be tuned. Parameters that are linearly related to the output may be optimally estimated by least-squares methods (LSE). One has to use non-linear optimization methods (like neural networks) for estimating non-linear mappings. [BVH99]

3 Model extraction approaches

3.1 Fuzzy model extraction methods

Fuzzy models have excellent capabilities to describe a given system. Many studies regarding fuzzy modeling have been reported [YF94]. Some of them are based on pattern-recognition [SY93, W94] and some others are based on system programming theory [TS85]. One of the most outstanding models among them is the model suggested by Takagi and Sugeno in 1985 [TS85]. However, this identification algorithm is too complex and difficult to implement. Lately, to solve this problem, Sugeno and Yasukawa proposed a new model based on pattern recognition techniques [SY93].

3.1.1 Takagi and Sugeno's Fuzzy Model

The fuzzy model suggested by Takagi and Sugeno [TS85] represents a mathematical tool, which is used to build a fuzzy model of a system. A fuzzy model of a non-linear system

consists of a set of implication rules, which are used to express control statements. An implication rule contains fuzzy variables with unimodal membership functions[1]. Since such membership functions are linguistically understandable, the fuzzy variables are also called linguistic variables. Takagi and Sugeno's fuzzy model approximates a nonlinear system with a combination of several linear systems by decomposing the input space into several subspaces and representing the input/output relationship, in each subspace, with a linear equation.

Let us assume a Multi- Input Single-Output (MISO) system with m inputs ($x_1, x_2, ..., x_m$) and a single output y. In principle, the fuzzy model of such a system consists of a rule base with n fuzzy implication rules. The i-th rule R^i ($i=1,2, ..., n$) has the following general form:

R^i: ***if*** $f(x_1 \text{ is } A^i_1,\ x_2 \text{ is } A^i_2,\ ...,\ x_m \text{ is } A^i_m)$ ***then*** $y^i = g^i(x_1, x_2, ..., x_m)$

Where,

y^i Inferred variable of the consequence of the i-th rule. The final output y of the system is a combination (a weighted average) of all y^i ($i=1,2, ..., n$)

x_k The *k-th* fuzzy variable of the premise ($k=1,2, ..., m$)

A_k The *k-th* fuzzy set whose membership function is a fuzzy subspace ($k=1,2, ..., m$)

f Connective function that joins the propositions in the premise.

g^i Function that implies y^i when the $x_1, x_2, ..., x_m$ satisfies the promise.

If f is the *"and"* connective function and g^i is a linear function of the form $a^i_0 + a^i_1 x_1 + ... + a^i_m x_m$ the i-th fuzzy implication rule becomes

R^i: ***if*** $x_1 \text{ is } A^i_1$ *and* $x_2 \text{ is } A^i_2$ *and ... and* $x_m \text{ is } A^i_m$ ***then*** $y^i = a^i_0 + a^i_1 x_1 + ... + a^i_m x_m$

The truth-value of the conjunction between propositions in the premise is estimated by the minimum of their membership values. That is, the truth value of (*x is A and y is B*) is estimated as *min(A(x), B(y))*, where *A(x)* and *B(y)* are the membership values.

Let y be the final output of the system and w^i be the truth value of $y=y^i$, then

w^i = (*truth value of the premise*) and (*truth value of the rule* R^i).

Assuming that the truth-value of R^i is *1*, then w^i is given by*

$$w^i = x_1 \text{ is } A^i_1 \text{ and } x_2 \text{ is } A^i_2 \text{ and } \ldots \text{ and } x_m \text{ is } A^i_m = \min\left(A^i_1(x_1), A^i_2(x_2), \ldots, A^i_m(x_m)\right) = \prod_{k=1}^{m} A^i_k(x_k)$$

[1] A membership function f is unimodal in *[a, b]* if there exist $t \in [a, b]$ such that, $f(x) \le f(y)$ for all $x \le y \le t$ and $f(x) \ge f(y)$ for all $t \le x \le y$. That is, f is monotonic in *[a,t]* and *[t,b]* with f(t) being the maximum value in *[a,b]*. Triangular, trapezoidal or bell typed membership functions are unimodal.

* The symbol $\prod_{i=1}^{m} a_i$ is used, in this context, to represent the $min(a_1, a_2, ..., a_m)$.

The truth value w^i is used as a weight to the consequence variable y^i to estimate the output y. That is

$$y = \frac{\sum_{i=1}^{n} w^i y^i}{\sum_{i=1}^{n} w^i}$$

The final output y is inferred from the n implication rules as the average of all y^i with the weights w^i.

The problem of system identification (extraction of fuzzy rules of the form R^i) requires the determination of the following items:

1. Identification of the premise variables (i.e. $x_1, x_2, ..., x_m$)
2. Identification of the membership functions of the fuzzy sets in the premise of each rule (i.e. $A^i_1, A^i_2, ..., A^i_m$, for $i=1,2, ..., n$). This is called *premise parameter* identification.
3. Identification of the parameters in the consequence of each rule. (i.e. $a^i_0, a^i_1, ..., a^i_m$ for $i=1,2, ..., n$).

Items 1 and 2 are related to the partition of the space of the input variables into some fuzzy subspaces, while item 3 describes the input-output relation in each subspace.

The identification of the fuzzy model is carried out iteratively as follows:

Assume some input variables and some initial premise parameters. Consequent parameters are optimally adjusted with the respect to the premise parameters and then the premise parameters are readjusted. This is accomplished by a complex algorithm, which is based on a non-linear optimization method. However, the implementation of this method seems to be difficult as pointed out by Wang and Langari [WL95], since it involves a non-linear optimization method.

The optimization of the fuzzy model is achieved by using a set of sample data (input/output examples). Let D and Y be the matrices containing r such input and output samples respectively. The *j-th* column of D yields the output y_j (i.e. $x_{1j}, x_{2j}, ..., x_{mj} \rightarrow y_j$).

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1r} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2r} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mj} & \cdots & x_{mr} \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 & y_2 & \cdots & y_j & \cdots & y_r \end{bmatrix}^T$$

3.1.1.1 Consequence parameter identification

Let us assume that we have n implication rules R^i ($i=1,2, ..., n$) of the form:

R^i: ***if*** x_1 *is* A^i_1 *and* x_2 *is* A^i_2 *and ... and* x_m *is* A^i_m ***then*** $y^i = a^i_0 + a^i_1 x_1 + ... + a^i_m x_m$

Given the matrices D and Y, the problem of consequence parameter identification concerns itself with the determination of the following parameter vector:

$$P = \begin{bmatrix} a^1_0 & \dots & a^n_0 & a^1_1 & \dots & a^n_1 & \dots & a^1_m & \dots & a^n_m \end{bmatrix}^T$$

Applying the *j-th* sample on the *i-th* rule R^i, we get the following

$R^i|_j$: ***if*** x_{1j} *is* A^i_{1j} *and* x_{2j} *is* A^i_{2j} *and ... and* x_{mj} *is* A^i_{mj} ***then*** $y^i_j = a^i_0 + a^i_1 x_{1j} + ... + a^i_m x_{mj}$

where $i=1,2, ..., n$ and $j=1,2, ..., r$

The weight of y^i_j is then given by

$$w^i_j = x_{1j} \text{ is } A^i_{1j} \text{ and } x_{2j} \text{ is } A^i_{2j} \text{ and } \dots \text{ and } x_{mj} \text{ is } A^i_{mj} = \min\left(A^i_{1j}(x_{1j}), A^i_{2j}(x_{2j}), \dots, A^i_{mj}(x_{mj})\right)$$

and the final output y_j is given by

$$y_j = \frac{\sum_{i=1}^{n} w^i_j y^i_j}{\sum_{i=1}^{n} w^i_j} = \sum_{i=1}^{n} \beta_{ij} y^i_j \quad , \text{ where } \quad \beta_{ij} = \frac{w^i_j}{\sum_{i=1}^{n} w^i_j}$$

Consequently, y_j may be expressed as

$$y_j = \begin{bmatrix} \beta_{1j} & \beta_{2j} & \dots & \beta_{nj} \end{bmatrix} \begin{bmatrix} y^1_j \\ y^2_j \\ \vdots \\ y^n_j \end{bmatrix} = \begin{bmatrix} \beta_{1j} & \beta_{2j} & \dots & \beta_{nj} \end{bmatrix} \begin{bmatrix} a^1_0 & a^1_1 & \dots & a^1_m \\ a^2_0 & a^2_1 & \dots & a^2_m \\ \vdots & \vdots & \vdots & \vdots \\ a^n_0 & a^n_1 & \dots & a^n_m \end{bmatrix} \begin{bmatrix} 1 \\ x_{1j} \\ \vdots \\ x_{mj} \end{bmatrix}$$

Therefore, the vector Y may be expressed as

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{21} & \dots & \beta_{n1} \\ \beta_{12} & \beta_{22} & \dots & \beta_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{1r} & \beta_{2r} & \dots & \beta_{nr} \end{bmatrix} \begin{bmatrix} a^1_0 & a^1_1 & \dots & a^1_m \\ a^2_0 & a^2_1 & \dots & a^2_m \\ \vdots & \vdots & \vdots & \vdots \\ a^n_0 & a^n_1 & \dots & a^n_m \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{12} & \dots & x_{1r} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mr} \end{bmatrix}$$

By reforming the matrices, Y may be written as

$$Y = X\,P$$

$$\text{where Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_r \end{bmatrix}, X = \begin{bmatrix} \beta_{11} & \dots & \beta_{n1} & x_{11}\beta_{11} & \dots & x_{11}\beta_{n1} & \cdots & x_{m1}\beta_{11} & \dots & x_{m1}\beta_{n1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \beta_{1r} & \dots & \beta_{nr} & x_{1r}\beta_{1r} & \dots & x_{1r}\beta_{nr} & & x_{mr}\beta_{1r} & \dots & x_{mr}\beta_{nr} \end{bmatrix} \text{and P} = \begin{bmatrix} a^1{}_0 \\ \vdots \\ a^n{}_0 \\ \dots \\ a^1{}_m \\ \vdots \\ a^n{}_m \end{bmatrix}$$

X is an $r \times n(m+1)$ matrix, Y is an r vector and P is an $n(m+1)$ vector. Therefore, the parameter vector P is calculated by

$$P = \left(X^T X\right)^{-1} X^T Y$$

The parameter vector P is recursively estimated by a stable-state Kalman filter by the following equations:

$$P_{j+1} = P_j + S_{j+1} X_{j+1} \left(y_{j+1} - X_{j+1} P_j\right)$$

$$S_{j+1} = S_j - \frac{S_j X_j + X_{j+1} S_j}{1 + X_{j+1} S_j \, X^T{}_{j+1}}, \qquad j = 0,1,\dots,r-1$$

$$P = P_r$$

With initial values $P_0 = 0$ and $S_0 = a\,I$, where a is a big number and I is the identity matrix. S is an $n(m+1) \times n(m+1)$ matrix, X_j is the *j-th* row in matrix X and y_j is the *j-th* element of Y.

3.1.1.2 Premise Parameters Identification

The premise parameter identification is concerned with the determination of the membership functions of the fuzzy sets in the premises. This is done by dividing the input space of each premise variable into fuzzy subspaces, provided that the premise variables are chosen.

The problem of finding the optimum premise parameters minimizing the performance index is reduced to a non-linear programming problem. For this purpose, the *complex* method is used. Each fuzzy set is represented by two numbers; one that gives the greatest grade 1, and the other that gives the least grade 0, since a membership function is linear.

3.1.1.3 Premise variable Identification

The proposed algorithm serves two purposes: (1) identifies the variables in the premises and (2) divides the variable space into several divisions. The number of the divisions must be also determined. This is a combinatorial problem.

Suppose we have m input variables $x_1, x_2, \dots, x_m$ and a single output y. The steps of the algorithm are as follows:

Step 1. The range of the variable x_1 is divided into two fuzzy subspaces *"big$_1$"* and *"small$_1$"* . The range of the other variables is not divided and therefore these variables do

not appear in the premise of a fuzzy rule. This means that the initial model consists of just 2 rules whose premises contain only the variable x_1. These rules are

if x_1 is big_1 **then** ...

if x_1 is $small_1$ **then** ...

Until we fix the final model, the algorithm produces many intermediate models in each stage. To keep track of these models, we name them as *"model j-i"*, which means the *i-th* model in the j-th stage. So, the above model is named as *"model 1-1"*. Similarly, we divide the range of all the remaining variable $x_2, ..., x_m$ and we then end up with m models, each of which is composed of two implications. Therefore the *"model 1-i"*, where $i=1,2, ..., m$, is of the following form:

if x_i is big_i **then** ...

if x_i is $small_i$ **then** ...

Step 2. For each *"model j-i"* the optimum premise parameters and consequence parameters are found by the algorithms described in the previous sections. The optimum model (the one with the least performance index) is chosen. This model is called the *stable state* model of the *j-th* stage. Suppose that the stable state model of the first stage is the *"model 1-k"*, where only the variable x_k appears in the premises.

Step 3. In this step, the premise variables of the stable state model are joined to each variable x_i ($i=1,2, ..., m$) by the *"and"* connective function forming, in that way, m models whose premises consist of the premise variables of the stable state model plus the variable x_i. For example, If the *"model 1-k"* is the stable state model of the first stage, we will have m joined variables of the form x_k-x_i ($i=1,2, ..., m$) in the second stage. Then, we divide again the range of each variable x_i ($i=1,2, ..., m$) into two parts (*small*, and *big*) and then we form m models (in the second stage) of the form:

if x_k is big_k and x_i is big_i **then** ...

if x_k is $small_k$ and x_i is $small_i$ **then** ...

if x_k is big_k and x_i is $small_i$ **then** ...

if x_k is $small_k$ and x_i is big_i **then** ...

The above model is called *"model 2-i",* where $i=1,2, ..., m$ but $i \neq k$. In the case of $i=k$, each range of the x_k is subdivided further into two additional parts forming, for example, *small*, *medium small*, *medium big*, *big*. Therefore, the implications of the *"model 2-k"* is as follows:

if x_k is $small_k$ **then** ...

if x_k is *medium* $small_k$ **then** ...

if x_k is *medium* big_k **then** ...

if x_k is big_k **then** ...

At this point, we perform step 2 in order to find which of the models *"models 2-i"* (*i=1,2, ..., m*) is optimum. The optimum one is chosen for the next stage.

Step 4. We repeat step 3 until the following criteria are satisfied

- The performance index of the stable state model becomes less than a predefined threshold.
- The number of implications of the stable state model exceed a predefined threshold.

In general each stage yields *m* models and only one (the optimum) is selected for the next stage. Each model in the *j-th* stage consists of 2^j implication rules. So, the predefined threshold about the number of implications must be a power of 2.

3.1.2 Sugeno and Yasukawa Model

The fuzzy model suggested by Sugeno and Yasukawa [SY93] consists of rules whose consequences are represented by linguistic variables which, effectively, results in the Mamdani model. This model is more intuitive than the one proposed in [TS85] and it is easier to implement. The *i-th* implication rule has the following form:

R^i: ***if*** x_1 *is* A^i_1 *and* x_2 *is* A^i_2 *and ... and* x_m *is* A^i_m ***then*** y *is* B^i

where $1 \le i \le n$ and x_k is the *k-th* input variable with $1 \le k \le m$. A^i_k, B^i are fuzzy variables of the premise and the consequence respectively. Let b^i be the crisp output of y in the *i-th* implication rule. Then, b^i may be obtained by defuzzing B^i by taking the center of gravity (the center of area of the consequent membership function B^i).

$$b^i = \frac{\int y B^i(y)\,dy}{\int B^i(y)\,dy}$$

The truth value of the premise w^i is again given by

$$w^i = x_1 \text{ is } A^i{}_1 \text{ and } x_2 \text{ is } A^i{}_2 \text{ and } \ldots \text{ and } x_m \text{ is } A^i{}_m = \min\left(A^i{}_1(x_1), A^i{}_2(x_2), \ldots, A^i{}_m(x_m)\right) = \prod_{k=1}^{m} A^i{}_k(x_k)$$

Therefore, the final inferred crisp output is estimated by taking the weighted average of the defuzzified values b^i with respect to w^i.

$$y = \frac{\sum_{i=1}^{n} w^i b^i}{\sum_{i=1}^{n} w^i}$$

It is often the case that we cannot build a fuzzy model over the whole input space because we lack data. In this case, there are some areas in the input space, which are not covered by the implication rules of the model. Consequently, there are some input data for which $w^i=0$ in all rules and y cannot be inferred from the model by using the above equation. In this case, we may infer the output y by using a *gradient* rule. A gradient rule may be used

to infer the output for a given input for which no rule is available and it has the following form:

$$R^i : \textbf{if } x_1 \text{ is } A^i{}_1 \text{ and} \ldots \text{and } x_m \text{ is } A^i{}_m \textbf{ then } y \text{ is } B^i \text{ and} \left(\frac{\partial y}{\partial x_1} \text{ is } C^i{}_1 \text{ and} \ldots \text{and } \frac{\partial y}{\partial x_m} \text{ is } C^i{}_m \right)$$

Where $\frac{\partial y}{\partial x_i}$ is the partial derivative of y with respect to x_i. The *core* area of such a rule, with m variables in the premise, is defined as an m-dimensional Cartesian space (having a membership function $A^i{}_k$ in each dimension), which is formed by the core elements of the fuzzy sets $A^i{}_k$ (those having membership value equal to one). The core area of the *i-th* rule is specified as follows: Let *core*($A^i{}_k$) is the core set of the $A^i{}_k$ premise parameter, then the Cartesian product *core*($A^i{}_1$) X *core*($A^i{}_2$) X...X *core*($A^i{}_m$) defines the core area of the rule. For example, If $A^i{}_k$ is trapezoidal and the premise consists of two variables, then the core space of the rule is a rectangle.

A gradient rule is much like an ordinary fuzzy rule, but it has an additional part with partial derivatives in the consequence. A model constructed by such rules is called a *gradient* model. The rationale beside using the gradient part in the consequence is to use a combination of some local fuzzy rules that might slightly match the input. For this purpose, the distance between the given input and the core region of a rule is used to determine which rules are able to contribute to reasoning. The reasoning algorithm of the gradient rules R^i is as follows:

1. Defuzzify B^i and $C^i{}_1, C^i{}_2, \ldots, C^i{}_m$ by taking the center of gravity. This yields the crisp values b^i and $c^i{}_1, c^i{}_2, \ldots, c^i{}_m$.

2. Calculate the distance d^i between the input and the core area of the rule. d^i may be defined as the Euclidean distance between the input and the point in the core area, which is closer to the input.

3. Then, the output y is inferred by

$$y = \frac{\sum_{i=1}^{n} \left\{ w(d^i) \left(b^i + \sum_{j=1}^{m} \left(d^i{}_j \quad c^i{}_j \right) \right) \right\}}{\sum_{i=1}^{n} w(d^i)}$$

Where $w(d^i) = e^{-d^i}$ is the weight of the *i-th* rule depending on the distance d^i. The term $b^i + \sum_{j=1}^{m} \left(d^i{}_j \quad c^i{}_j \right)$ is the extrapolated value of the output using its partial derivatives $\partial y / \partial x_j$.

3.1.3 Pros and Cons

Takagi and Sugeno's model can express a highly nonlinear functional relation using small number of fuzzy rules. However, the complexity of its identification procedures made it difficult to be used. In Sugeno and Yasukawa's model, the identification algorithm is sim-

ple. However, because the model uses singletons as consequent parts, it requires many fuzzy rules and its capability of system description is poor.

3.2 Neurofuzzy model extraction methods

It is generally agreed that fuzzy inference systems (FIS) provide a useful way of representing human knowledge in a fairly readable way in form of fuzzy inference rules. FIS rules are also capable of representing inexact knowledge and to reason with such knowledge in a theoretically sound way. However, the tuning of FIS proves to be challenging, as in a nontrivial FIS there are quite a few parameters to modify (typically the membership function parameters). It would be useful to be able to create or tune a FIS based on a training data set of input values and the desired target outputs. One of the ideas has been to apply the learning abilities available with the neural network architectures to the tuning of FIS.

In the neural network domain supervised learning task is often solved using a feed-forward layered network structure with simple processing units organized in layers. The nodes in each of the layers are typically fully connected with those of the neighboring layers. For each of the nodes there are typically only few adjustable parameters like the weights from each of the neighbors and a bias weight. The network is adjusted to a set of learning data (inputs and outputs) by feeding the inputs into the system, propagating the evidence through the network and by calculating the difference from the desired target. Then the parameters are adjusted by gradient descent optimization performed by a special error back-propagation algorithm.

Jang [Jan93] has introduced the ANFIS architecture (Adaptive Network based Fuzzy Inference System). Figure 3-1 provides an example of a simple FIS represented in an ANFIS network. In ANFIS architecture, a FIS is described in a layered, feed-forward network structure where some of the parameters are represented by adjustable nodes (represented as rectangular entities in the figure) and the others as fixed nodes (represented as spherical entities in the figure). The raw inputs are fed into the layer 1 nodes that represent the membership functions. The parameters in this layer are called premise parameters and they are adjustable. The second layer represents the T-norm operators that combine the possible input membership grades in order to compute the firing strength of the rule. At least in the basic ANFIS method these parameters are not adjustable. The third layer implements a normalization function to the firing strengths producing normalized firing strengths. The fourth layer represents the consequent parameters that are adjustable. The fifth layer represents the aggregation of the outputs performed by weighted summation. It is not adjustable.

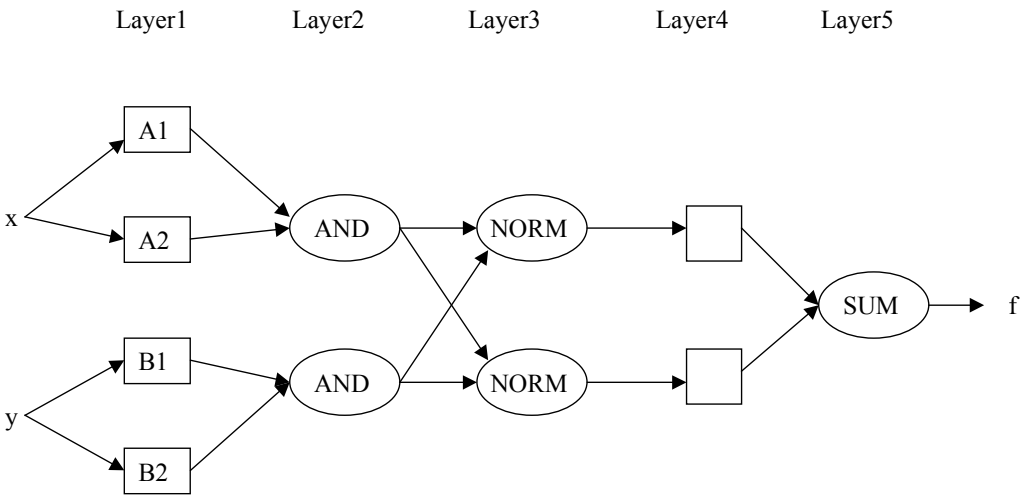


Figure 3-1. An ANFIS network structure for a simple FIS.

Jang introduces a two-pass algorithm for adjusting the parameters using a modified error-backpropagation optimization algorithm. In the forward pass the premise parameters are held fixed and the consequent parameters are adjusted by least squares estimation (LSE). In the backward pass the network error is backpropagated through the network and the premise parameters are adjusted by gradient descent while the consequent parameters are held fixed.

The basic ANFIS method does FIS parameter adjustment to a predefined model. It may be a hand-tuned system or automatically generated. Only the membership function parameters are adjusted. A suitable initialization to the membership functions allocates membership functions for the whole range and would make the boundaries overlap somewhat so covering the whole input range.

The inputs are not automatically selected nor the general network parameters adjusted. The variable selection may be supported by some type of searching (exhaustive in simple cases, directed forward searching, heuristic or a genetic one). Also the network parameters (like the T-norm functions to use, number of membership functions to use) could be adjusted by a supervisory control system.

In [Jan93] multiple examples of the ANFIS were provided ranging from nonlinear regression on a few inputs to time series prediction of a chaotic time series. The results were reported to be comparable with neural network ones. However the tests were done on noiseless data sets which were generated by a clear functional pattern. So the operation on noisy data was not proven.

4 Application case study

Our research problem concerns the detection of abnormalities in a process control environment as early as possible. In this report we study the applicability of automated knowledge discovery methods for building prediction models based on process measure-

ment databases. Especially fuzzy model extraction methods are studied in this report. The problem may be approached using different methods ranging from building monitoring models capable of detecting abnormalities in the measurement data records to temporal prediction models capable of forecasting the future behavior of quality indicator values ahead of time. One of the main motivations is to be able to predict abnormal situations in advance so that corrective actions can be taken in time. The prediction task is considered more thoroughly in the case of an activated sludge waste water cleaning system.

4.1 Description of the activated sludge waste water cleaning process

Our test case, called WasteWater, concerns two biological subprocesses of an activated sludge waste water cleaning system being used at the Kirkniemi factory of Metsä-Serla, a Finnish wood-processing company. Here we shall briefly describe the relevant aspects of the process being studied.

Activated sludge waste water cleaning is a complex process consisting of mechanical, biological and chemical subprocesses. The waste water is continuously fed to the process and then it flows through the process for days. The considered process consists of two partly connected cleaning lines both containing an aeration basin and a sedimentation basin. Before entering these lines, the waste water has been mechanically cleaned, its acidity has been neutralized and additional nutritions have been added to the system. The cleaning is based on allowing the bacteria, protozoans and other micro-organisms to eat organic waste as nutrition.

In the aeration basins additional oxygen is dissolved into the water so as to keep the process aerobic, and thus to allow and accelerate the purifying bacterial consumption process. The micro-organisms and the organic waste form a biosludge, which is extracted from the water in the sedimentation phase. A selected part of the sedimented sludge is circulated through the process back to the aeration basin to be reactivated in order to reach a predefined sludge age (meaning the average amount of time that a particle is kept in the circulation). To maintain stability of the process and good quality of the sludge, excess sludge is periodically removed from the process and dried. Removing the sludge too soon may produce too much waste to be dried and to be transported to a dumping place. Cycling the waste water too long in the system consumes excess cleaning capacity.

The circulation process is slow; a typical delay between the beginning and the end of the circulation process is 2-3 days. Furthermore, the biological aspects of the process have longer delays, typically 10-15 days. An overview of the considered process is presented in Figure 4-1.

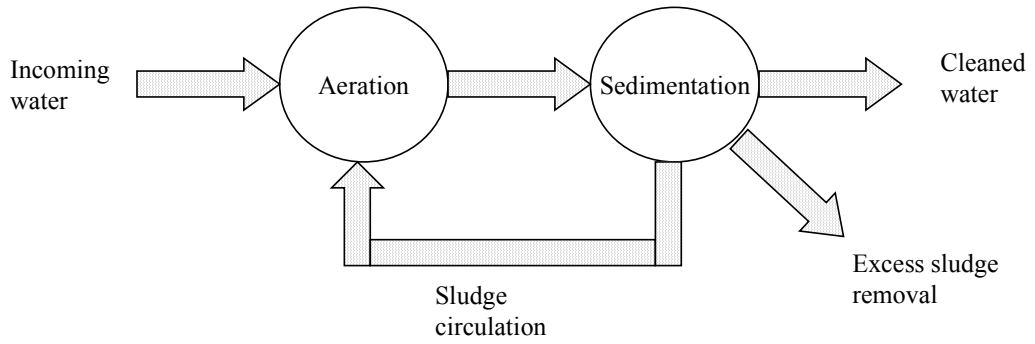


Figure 4-1. Biological waste water cleaning process.

The process operators control the process on the basis of process measurements, such as BOD[2] and COD[3] levels, and prior knowledge. In normal situations, when quantity and quality of the incoming water is stable, control of the process is easy and well known. In abnormal situations, for example, when chemical concentrations of the input water change rapidly, the process can go out of balance. The biomass reacts slowly to changes in the water quality and quantity, and therefore an out-of-balance situation in the waste water treatment plant occurs several days after the actual cause has occurred. Furthermore, corrective actions are also slow and the results are usually observable only after quite a long delay.

Process measurements are of two basic types: online meterings and laboratory analysis. Online meterings are obtained using sensors and the values from online meterings are available instantly, together with a timestamp. Sometimes online meterings may produce erroneous indications. This is due to fouling of the sensors and drifting of sensor calibration. Typically values from online meterings are available with a time resolution of several seconds, but in this case study averaged values of one hour are used.

Most process quality results are obtained using laboratory analysis. Analysis are performed by taking a sample which is then analyzed in laboratory by chemical experts. Results of the laboratory analysis are entered to the computer system manually with a timestamp coding the time instance the sample was taken. Most of the laboratory analyses are performed in-house, but some analyses are performed in external neutral laboratories. Typically, results of in-house analyses are available within hours (by late afternoon) after the sample period has ended, but external laboratory results can have a delay of several weeks. The sample period may extend for several days.

---

[2] The biological demand for oxygen (BOD) is a measure of the amount of oxygen used by microorganisms to decompose the organic matter in the wastewater.

[3] The chemical demand for oxygen (COD) is a measure of the amount of oxygen used to oxidize organic matter and to convert it to carbon dioxide and water.

4.2 Problem formulation

The problem of detecting abnormal situations may be formulated in different manners motivating alternative approaches to the problem:

- One approach is to try to forecast the continuous-valued outputs of the time series variables a few time steps into the future using some type of regression methods.

- Often in process environments normal behavior clearly dominates the available process data meaning that examples of abnormal situations is much more rare. Here one may try to model the normal behavior of the system by clustering the measurement data records into similar prototype vectors, which then describe the typical states of the system. Then one may monitor the distance of each new state from the closest prototype and if the distance is above a given threshold value, an abnormal state has been identified and more detailed analysis methods may be started. [IRV97].

- Another approach is to build a classifier assigning each of the process conditions into typical process states based on the historical measurement records available at the time of the classification task. This way the system could warn the user about potential problems before the quality indicators show problems. Examples of good and bad behavior are needed for building classifiers. The state labels could be normal or abnormal states identified based on the value of the desired quality indicators after the prediction time interval has passed (future values). [SVAH99].

4.3 Measurement database and the preprocessing methods

Our measurement database contains daily values for 338 consecutive days. Values are available for 17 original variables (online, laboratory and calculated ones). The time series of these variables have been cleaned by removing clear outliers or erroneous values and by replacing missing values using interpolation. The time series have also been smoothed.

The temporal aspects of the data set have been captured by adding additional features to each record. A wavelet transformation was calculated for each series and four parameters describing the signal with different temporal detail are stored as new features (W1 – W4 prefixes in the field names). W1 has the lowest temporal detail and W4 is the most precise in temporal sense. A different way of viewing the temporal variations is to study the differences of values between timestamps. Three levels of differences were computed for each variable (D1, D3 and D7 prefixes, the number specifies how many days separate the compared timestamps). These steps added 7 * 17 new feature variables into the record. Therefore we have 136 input fields and additionally the output fields.

Various other ways of generating meaningful temporal features are available but they have not been tested here.

4.4 Experiences with Matlab Fuzzy Logic Toolbox

4.4.1 Main features of the Fuzzy Logic Toolbox

The Fuzzy Logic Toolbox [Mat00] is a library of functions implementing a framework for creating, editing and executing fuzzy inference systems. It is based on the Matlab mathematical programming environment by the Mathworks company (http://www.mathworks.com). Both graphical tools and command line scripts are available. Both fuzzy clustering and adaptive neurofuzzy techniques (based on the ANFIS architecture) are available for generating models directly from training data. The created systems may be distributed as standalone fuzzy engines. Matlab allows the programmer the possibility to also integrate other toolbox functionality and advanced plotting and graphics functionality into the user M-files. The open programming interface and the ability to extend the tools is one of the main benefits of the Matlab environment. For example, it is possible to test fuzzy controllers directly in a simulator created with Matlab/Simulink.

The toolbox allows one to build fuzzy inference systems using both Mamdani and Takagi-Sugeno methods. The Mamdani method is the more intuitive and well-suited to human input and also the more widespread method. The Sugeno method is computationally more efficient and well suited to optimization and adaptive techniques and to mathematical analysis. It has also guaranteed continuity of the output space. The methods are very similar as the fuzzification of the inputs and the application of the fuzzy operator are similar. However, in Sugeno method the output membership functions may only be constant or linear. A typical first-order Sugeno Fuzzy model has rules of the form:

if x is A_i and y is B_i then $z = p*x + q * y + r$

Therefore the implication method and the aggregation methods are different from the Mamdani way.

The toolbox provides three types of graphical editors: the FIS Editor for editing the main structure of the fuzzy inference system by linking the fuzzy inputs and outputs to the system and specifying the main parameters of the system. It seems that only simple one-layer models may be defined and combining different rule sets is not allowed. The Membership Function Editor is used for specifying the membership function parameters. There are 11 built-in membership functions (user may define additional ones). The rule editor is used to link the different input combinations to the outputs. The Rule Viewer may be used to graphically inspect the rules. The Surface Viewer is used to display the response surface of the system.

The generated fuzzy inference systems may be stored in an open ASCII file format.

Fuzzy clustering is also supported. The purpose of clustering is to identify natural groupings of data from a large data set. Fuzzy c-means clustering (FCM) is a clustering method where each data point may partially belong to more than one cluster with a degree specified by a membership function. FCM starts with an initial guess for the cluster center locations and iteratively updates the cluster centers and the membership grades for each data point based on minimizing a cost function. The algorithm typically converges within a few iterations. In Figure 4-2 one can see a simple example of fuzzy clustering. There a set of two dimensional data points has been clustered with two clusters and the cluster

centers have been plotted as well as the data points colored based on the cluster that they most likely belong to.

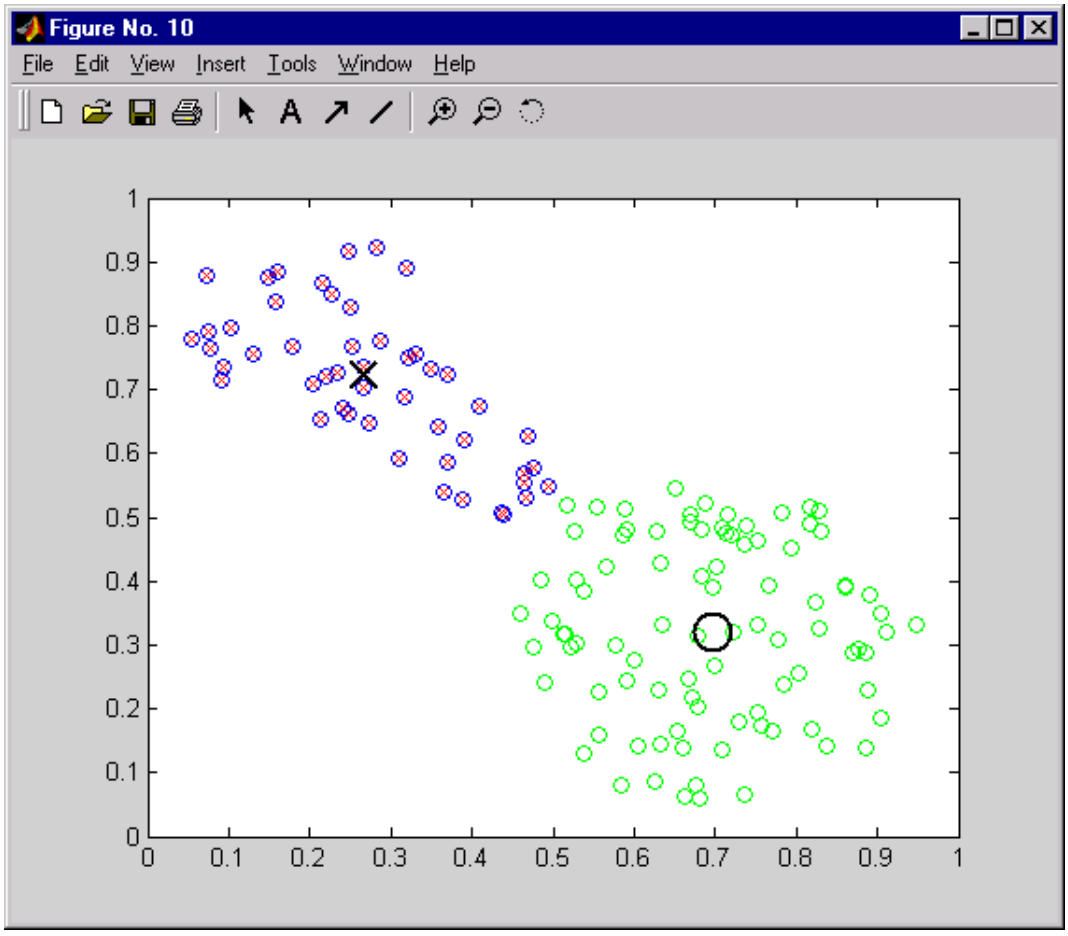


Figure 4-2. An example of finding two clusters for a 2-dimensional data set

Subtractive clustering method is a fast, one-pass algorithm for estimating the number of clusters. It partitions the data into clusters and generates a FIS with the minimum number of rules required to distinguish the fuzzy qualities associated with each of the clusters.

Adaptive neurofuzzy learning may be used when one has the input/output data for the system one wants to model. ANFIS computes such membership function parameters that best match the given data. The generated models are limited to the Sugeno type models (the 0th or the first order) with a single output and only predefined membership function formats may be used. The algorithm requires an initial FIS structure specifying the number and type of membership functions. These may be created manually, by using grid partitioning or by subtractive fuzzy clustering. Also a graphical user interface, called ANFISEDIT, is available for using the ANFIS functionality.

4.4.2 Case test 1: Prediction of future values of a quality variable

We tested the ANFIS learning in the problem of predicting future values (3 days forward) of quality variable LiukP3_4 based on any of the features available at the prediction time.

The quality variable values below 0.5 are considered normal by the operator. The operator is most interested in learning accurate predictions about abnormally high values.

The data set was augmented by adding new features to each data record representing the lagged values of Liuk_P3 variable (3, 6, 9 and 12 days before the prediction time). This was done to test the autocorrelation of the predicted time series.

The available data set was divided into three sets called training, testing and validation data sets. Samples were assigned alternately to the data sets and therefore each of the data sets were allocated about 1/3 of the samples. As the value ranges of the variables are very different it was necessary to normalize the value ranges before performing the modeling. Each data set was (0,1)-normalized (mean=0 and standard deviation=1) based on the training data set. Also all the data that is used for making predictions has to be scaled before feeding into the model. The predictions are then denormalized into the real value ranges.

As there is not much training data (131 records) one must not build too complex models as that leads to poor generalization capability. Also fuzzy rule sets are best suited to mapping few inputs to one or more outputs. Therefore a suitable model was searched for by first using each of the available input features alone as the predicting variable in the generated FIS model and by testing each of these models on their RMS error on unseen checking data. The models were ranked based on this check RMS error. Then either an exhaustive search or a sequential forward search may be done. In the former all the combinations of two or more inputs are tested and the best one is selected based on the check RMS error. In the latter a directed beam search is performed where one additional input is added to the best fitting model (measured with check error). It is clear that the exhaustive search soon becomes too complex and sequential forward search is therefore a more suitable alternative.

Using sequential forward search for searching two input variable models and 5 membership functions per variable it turned out that W2LIUK_P3 (2nd wavelet coefficient of the values of variable LIUK_P3) produced the lowest RMS error (about 0.78) for the checking data. However, the differences between the models were not very large (see Figure 4-3). It is noticeable that all of the two variable models produced larger checking errors than the best one variable models although their training errors were smaller (therefore overfitting occurred).

In Figure 4-4 one sees the predictions for the validation data set. Some of the peaks have been anticipated but there are two sets of very large negative peaks. Possibly the negative estimates could just be ignored but anyhow the predictions do not anticipate well the actual values. Similar results have been found also with neural networks using the same data.

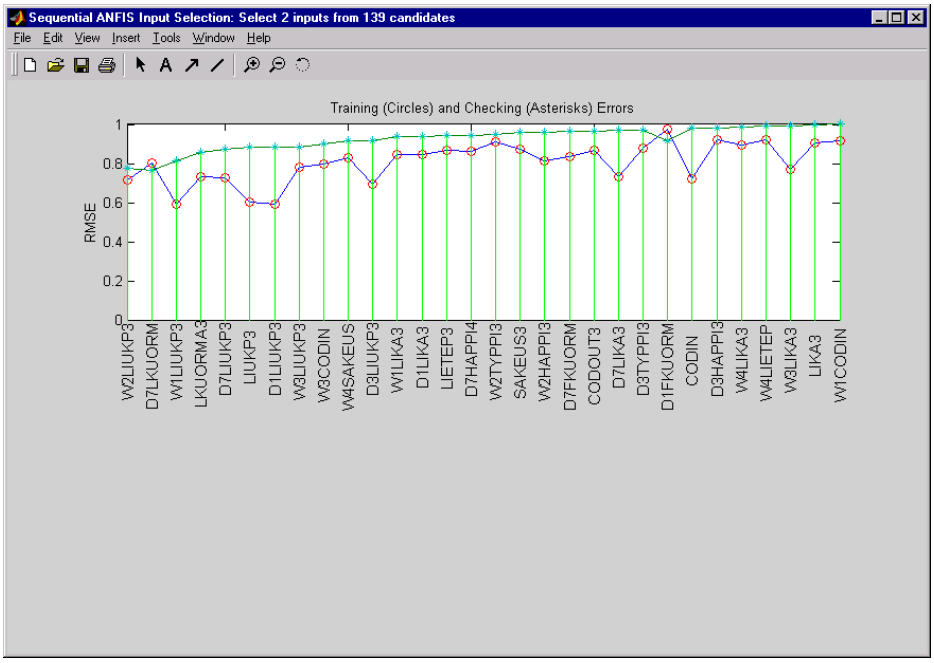


Figure 4-3. Checking and training RMS errors for the 30 best one- or two-variable models using 5 membership functions per variable

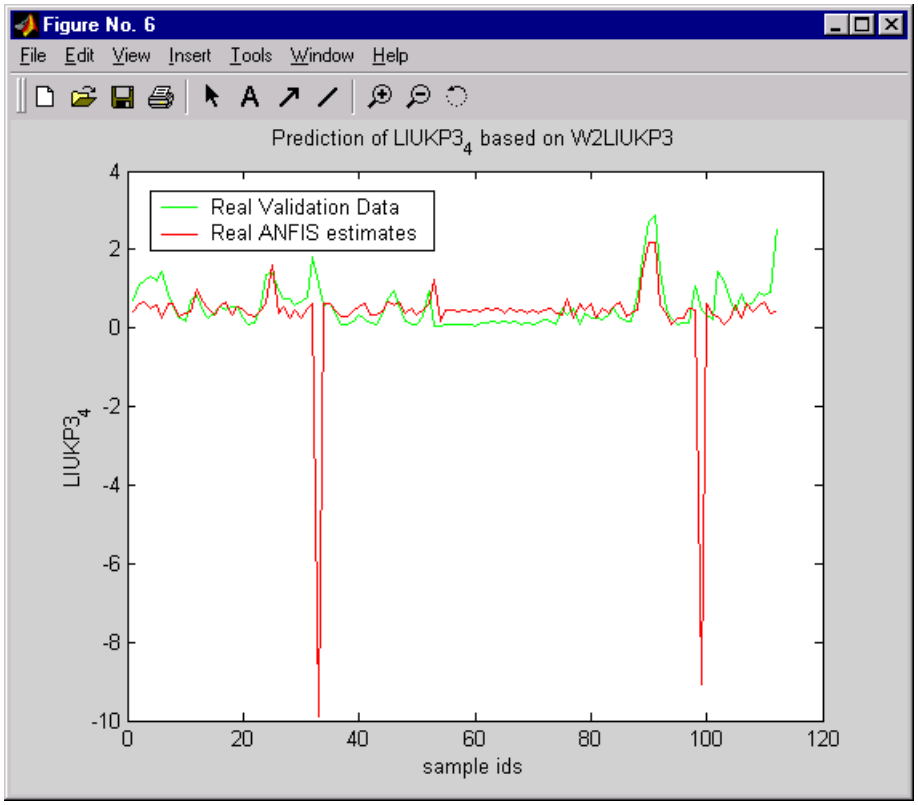


Figure 4-4. ANFIS predictions (red/dark) and the real outputs (green/light) for a validation data set based on a one input (W2LIUKP3) fuzzy model with 5 membership functions per variable

It was found that the time delayed features of Liuk_P3 did not produce good results suggesting that there is only weak autocorrelation in this time series.

The effect of the number of membership functions (MF) per variable was tested by creating models using only three MFs per variable. It was found that with these simpler models also two variable models faired better on the checking data (with training data the fitting was much better). Now the VIRTA3 variable produced best models (see Figure 4-6). The differences are however small between the models. From Figure 4-7, we see that now the large negative estimates are missing. However, many of the peaks remain unpredicted and some false ones are suggested. From Figure 4-8 and Figure 4-9 one sees that even with the training and checking data the produced estimates are not very good suggesting that overfitting is not present and that the one-variable model is not enough to predict the phenomena described in the data.

The generated rule bases contained as many rules as in the Cartesian product of the membership function amounts of each of the inputs. So as the number of inputs rises the number of rules soon becomes untractable. The number of rules depends on the number of input variables and on the number of membership functions used per variable (see Figure 4-5 for a few examples)

#rules = (#vars) $^{\#MF}$

#variables	#MFs	#rules
2	3	8
3	3	27
4	3	64
2	5	32
3	5	243
4	5	1024

Figure 4-5. The number of generated rules for a few cases-.

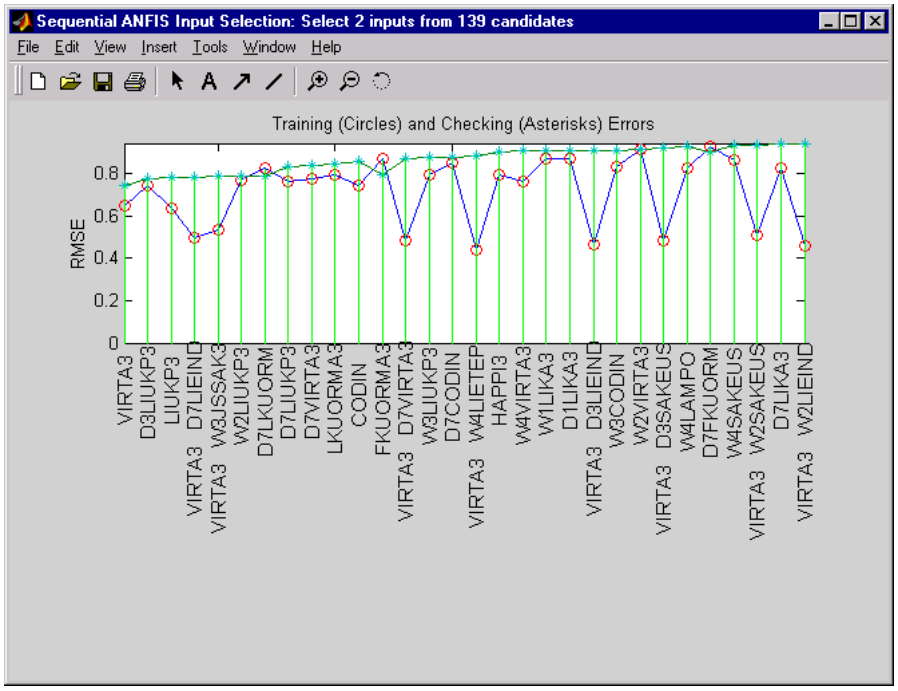


Figure 4-6. Checking and training RMS errors for the 30 best one- or two-variable models using 3 membership functions per variable

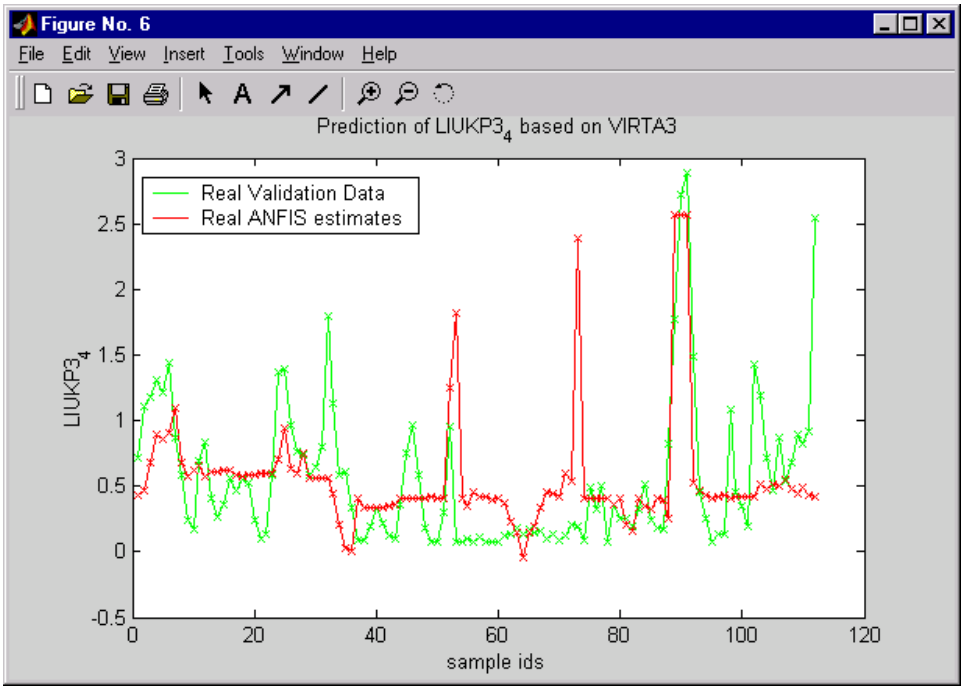


Figure 4-7. ANFIS predictions (red/dark) and the real outputs (green/light) for a validation data set based on a one input (VIRTA3) fuzzy model with 3 membership functions per variable

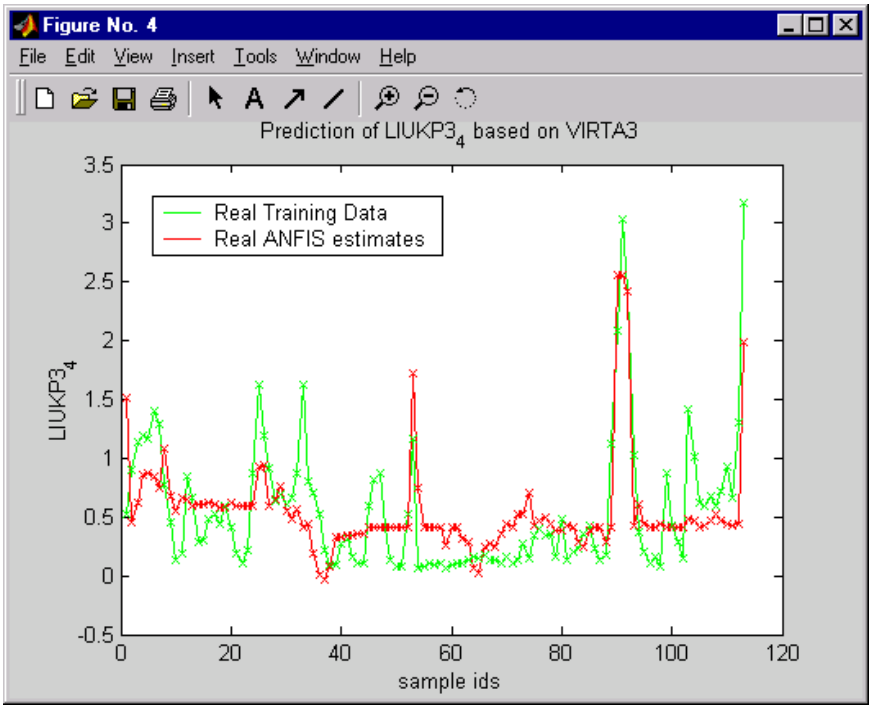


Figure 4-8. ANFIS predictions (red/dark) and the real outputs (green/light) for the training data set based on a one input (VIRTA3) fuzzy model with 3 membership functions per variable

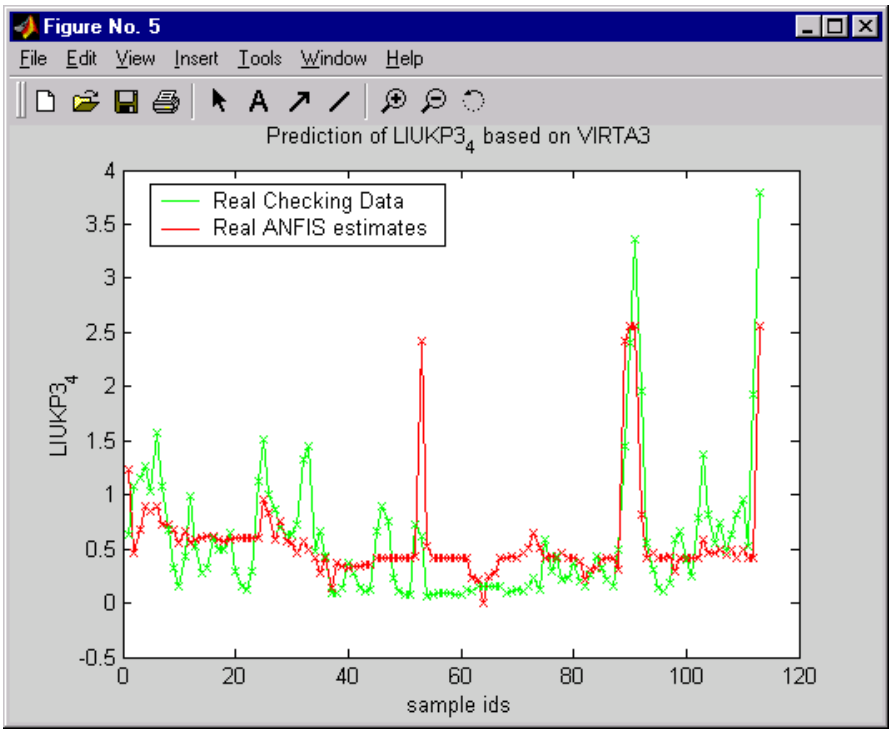


Figure 4-9. ANFIS predictions (red/dark) and the real outputs (green/light) for the checking data set based on a one input (VIRTA3) fuzzy model with 3 membership functions per variable

4.4.3 Case test 2: Fuzzy clustering of the input data records

Fuzzy clustering of the measurement vectors was tested by clustering the training data using subtractive clustering. At first a suitable number of rules was determined and then linear least squares estimation was used to determine each rule's consequent equations. A fuzzy rule set was then created to map the clusters to the desired outputs. In Figure 4-9 we see that the predictions seem to anticipate the actual behavior fairly well for unseen validation data set. However, this may be due to the fact that the data sets were assigned data for consequent days. Therefore, the data records are not completely independent. Also one must note that the generated FIS contained 20 inputs and for each input 63 membership functions (63 rules were generated, there were 112 training data records) which means that the model is much too complex and the good results are due to data memorization.

The toolbox also contains facilities to do fuzzy c-means clustering which iteratively calculates the requested amount of cluster centers. The tool was tested and worked fast. However, the quality of the clusters was not evaluated, as no easy way was available.

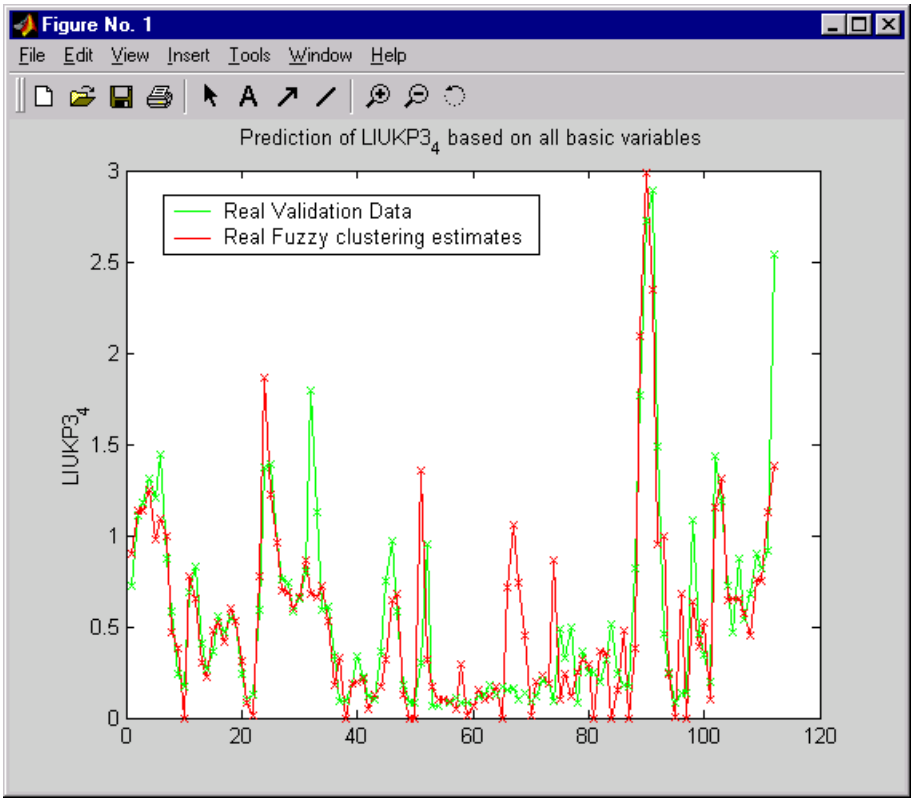


Figure 4-9. Fuzzy clustering based predictions (red/dark) and the real outputs (green/light) for the validation data set based on all basic variables (20).

4.5 Experiences with fuzzyTech

fuzzyTech by Inform GmbH is a tool for the development of fuzzy logic and neural-fuzzy solutions. We tested it with the WasteWater case data already described in this document.

The main problem with the evaluation of the software was a lack of comprehensive documentation. For example, the method used for neural-fuzzy learning was not revealed for the user.

The fuzzy models constructed in the test runs were solid and understandable. The predictability of the models was not tested with RapidBase so the value of the neurofuzzy learning module was not evaluated. For more about the fuzzyTech – RapidBase interoperability, see Section 5 "Running generated models with RapidBase".

Based on the test period with the product a major drawback seems to be the restriction of the rule base size to no more than 1000 rules. This limitation restricts the number of input/output parameters of the model because of the following reason: The initial model is formed by issuing a rule for each fuzzy term combination of each input and output variable. This can be seen as a Cartesian product of all the variables and terms involved. For example, if we have a model formed by three input variables and one output variable, each having four linguistic terms then the size of the initial rule base is going to be $4^{(3+1)} = 256$ rules.

The rest of the Section shows how the fuzzy type and rule base modifications are handled in fuzzyTech. In addition to manual modifications also the neurofuzzy learning module is addressed.

4.5.1 The user interface

The user interface (UI) of fuzzyTech follows the familiar look&feel of the latest generation of MS Window 2000 and MS Office 2000 software products.

The tree view pane enables structured access to all components of a fuzzy logic system under design in the same way the Windows Explorer lets users browse the structure of their PCs. The Editor and Analyzer windows allow designing each single component of a fuzzy logic system graphically. The following Figure gives a view of the graphical UI.

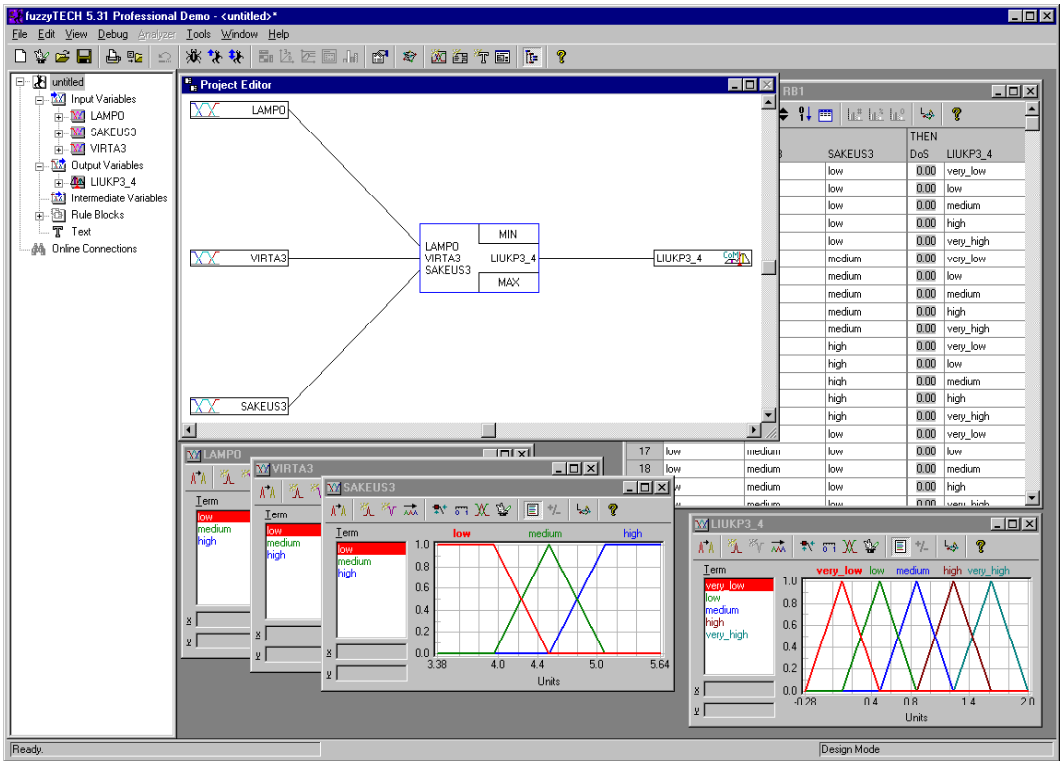


Figure 4-10. An overview of the user interface.

The project open on the Figure above was generated from an example data stored in a text file. The system automatically generated the input variables, their linguistic types, initial rule base and the output variable. The definition of a linguistic type can be easily changed in the linguistic type window (distinct for each type defined) using common mouse techniques (e.g., dragging). The following Figure shows an example of a linguistic type window.

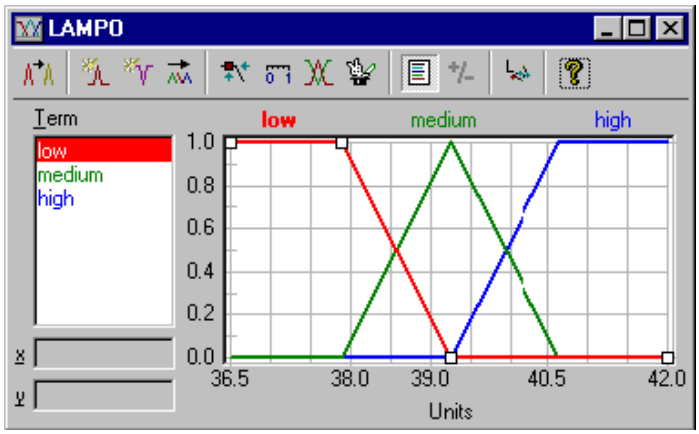
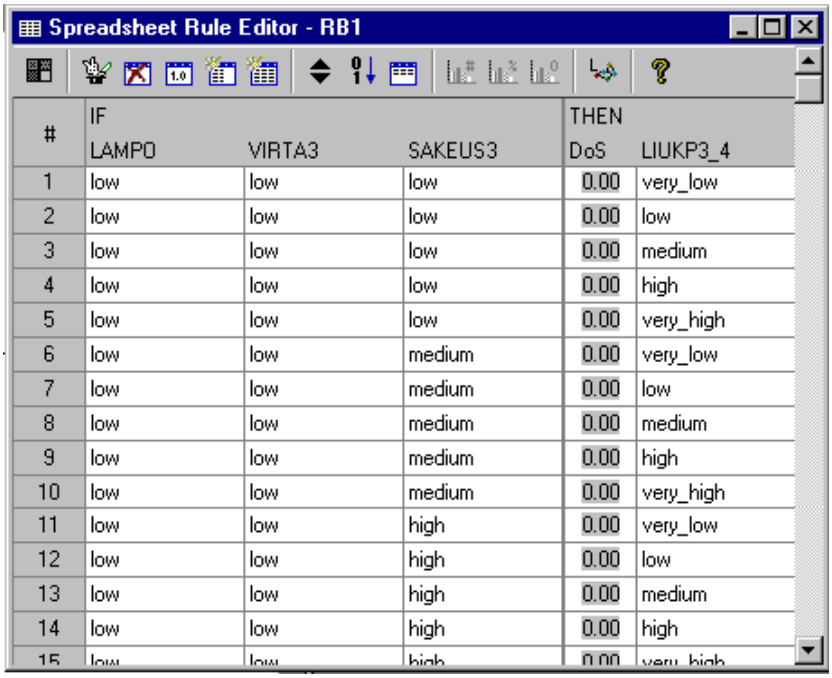


Figure 4-11. A linguistic type.

fuzzyTech uses Mamdani type rules to form the rule base (IF fuzzy antecedent THEN fuzzy consequent). In addition, the Degree of Support (DoS) value is introduced with

every rule. A DoS value gives a weight for each value to be used in the rule aggregation step of fuzzy inference. The value is between [0...1]. Effectively, the DoS value may be used as a way for structural learning of the fuzzy model. Using the value one can limit the size of the rule base, for example, by dropping off the rules with the DoS value less than a specific value. A part of the rule base for our example is shown below.



#	IF			THEN	
	LAMPO	VIRTA3	SAKEUS3	DoS	LIUKP3_4
1	low	low	low	0.00	very_low
2	low	low	low	0.00	low
3	low	low	low	0.00	medium
4	low	low	low	0.00	high
5	low	low	low	0.00	very_high
6	low	low	medium	0.00	very_low
7	low	low	medium	0.00	low
8	low	low	medium	0.00	medium
9	low	low	medium	0.00	high
10	low	low	medium	0.00	very_high
11	low	low	high	0.00	very_low
12	low	low	high	0.00	low
13	low	low	high	0.00	medium
14	low	low	high	0.00	high
15	low	low	high	0.00	very_high

Figure 4-12. A rule base.

The neurofuzzy learning tool provides a graphical interface for the learning process monitoring. This interface is interesting to use and gives information about the values of the learning parameters. The drawback of this tool is the fact that it slows down the learning process considerably. The learning process can be executed without the graphical user interface thus speeding up the process. The learning module can learn Dos values of the rules or the definitions of the linguistic types or both at the same time. The UI of the tool can be seen in the following figure.

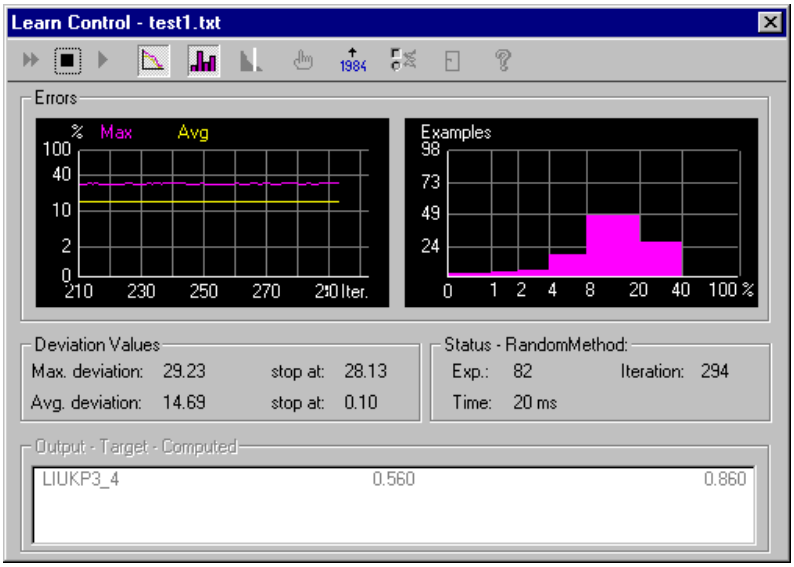


Figure 4-13. The learning tool.

The on-line learning progress information on the window above include error percentage of the current rule base against the learning examples. The maximum deviation (the red curve on the left) shows the error of the worst sample against the current rule base. Samples under the threshold are skipped within the training procedure. Training one sample can increase the error of other samples. Therefore, the training results are automatically performed after every complete iteration. The average deviation (the green curve on the left) shows the average of the errors occurring during a complete iteration. The average error is computed by adding the errors for each sample divided by the number of samples. Also, the process status in terms of the number of iterations on the test examples and the elapsed time are printed.

4.5.2 Under the hood

Unfortunately, the actual learning method of the fuzzy model is not known. Instead of introducing the learning method we are restricted to explain the parameters the user can adjust to control the process.

When the example data is fed to the system it recognizes the number of variables *n* and by default suggests *n-1* input variables and one output variable. It also automatically forms the definition area for each variable and assigns default linguistic terms for each variable. This process is interactive and the default settings can be changed by the user.

Before applying the neurofuzzy learning module the learning process parameters have to be set up (if not, the defaults are in effect). The following are the basic user defined parameters: *Step Width (DoS)* that sets the value for the learning rate used for updating the DoS of fuzzy rules and *Step Width (Term)* that sets the value for the learn rate used for updating the position of terms of linguistic variables.

The stop conditions for the learning process are set by the user. One of the following can be selected: *Max Steps* that stops the training after a fixed number of iterations. The *Avg Deviation* criteria is fulfilled if the average of the errors occurring during a complete

iteration is less than an error threshold. The average error is computed by adding the errors for each sample divided by the number of samples. The *Max Deviation* criteria compares the error of the worst sample with an error threshold. Samples under the threshold are skipped within the training procedure.

The user has to select also the learning method. It can be either *RealMethod*, *RandomMethod*, *Batch_Learn* or *Batch_Random*.

- RealMethod uses a single selected sample to find the best terms and rules to be changed. The changes to membership functions and fuzzy rules are computed by using the constant StepWidth(Term) to change terms, and the constant StepWidth(DoS) to change rules.

- RandomMethod is like RealMethod but only using random steps from the equipartioned interval [0…StepWidth(LingVar)] to change terms, and random steps from the equipartioned interval [0…StepWidth(DoS)] to change rules.

- Batch_Learn computes a batch in which all samples are used to find the best terms and rules to be changed. The changes to membership functions and fuzzy rules are computed by using the constant StepWidth(Term) to change terms, and the constant StepWidth(DoS) to change rules.

- Batch_Random is like Batch_Learn but only using random steps from the equipartioned interval [0…StepWidth(LV)] to change terms, and random steps from the equipartioned interval [0…StepWidth(DoS)] to change rules.

The neurofuzzy training is based on sample data. The training success depends on the order in which the samples are selected. The user selects the selection mode that determines whether samples are used sequentially from a file or if they are used in random.

4.6 Summary of the application case experiences

In this study the WasteWater measurement time series database has provided the test bench for experimenting with automated tools for constructing fuzzy inference systems. Most experiments were performed using Matlab Fuzzy Logic Toolbox with a full evaluation copy. The aim was to explore what is possible with commercially available tools.

Preprocessing, division to different data sets (training/testing and validation) and complexity control of the generated models were considered. The generated models were evaluated on unseen validation data set. Sequential forward selection was found useful for selecting the input variables to include into the model. New inputs were added to the model only if they could produce better fitting models for separate testing data set. ANFIS method was used for tuning the parameters in the FIS models.

Experiments were also performed using the fuzzyTech tool. The tested demo version was, however, strongly limited in the size of models that could be evaluated (number of data samples and the maximum number of input variables were strongly limited).

5 Running generated fuzzy models with RapidBase

In this Section we consider the applicability of studied methods to RapidBase. The following model depicts the overall process combining the commercial fuzzy tool and RapidBase.

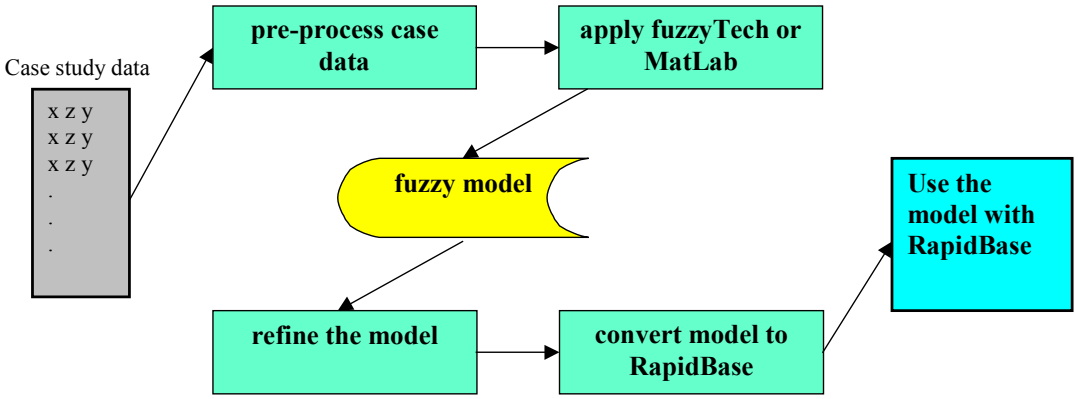


Figure 5-1. The overall process.

In the sequel, both Matlab and fuzzyTech applicability with RapidBase is addressed in the subsections that follow.

5.1.1 Matlab

The Fuzzy Toolbox of Matlab produces Takagi-Sugeno type fuzzy models. This already poses a problem for the model applicability in RapidBase cause the fuzzy model used in RapidBase is of the Mamdani type. In addition, the other methods used in fuzzy inference processing, like rule antecedent proposition combination method and the rule implication method offer more flexibility in Matlab than RapidBase can currently handle.

In order for RapidBase to support Matlab generated fuzzy models, at least the following additional feature has to be implemented in RapidBase:

- Tagaki-Sugeno inference method (currently only Mamdani)

In addition, a program has to be implemented to interpret and convert the Matlab output file format to proper Rapidbase RQL file.

5.1.2 fuzzyTech

fuzzyTech produces Mamdani style fuzzy models and thus is more straightforward than Matlab to use with RapidBase. fuzzyTech uses the DoS (Degree of Support) values for rules to adapt the fuzzy model to the set of teaching examples. The DoS functionality was implemented in RapidBase during the course of the Fume project to support fuzzyTech models.

Still, a program is needed to interpret and convert the fuzzyTech output file format to proper Rapidbase RQL file.

6 Conclusions

We have analyzed possibilities of automatic extraction of fuzzy inference (FIS) models from industrial observation data. Theoretical approaches to identification of both the Mamdani and Takagi-Sugeno models has been studies and limitations thereof analyzed. The second step of model extraction—the parameter identification—turns out to be feasible to be performed automatically using various methods. However, the first step—the structural identification—is demanding and typically requires some human input. The structural identification includes choices of rule types and selection of input and output variables. There are some auxiliary methods, like fuzzy clustering of input space, that may be of some help but they have to be carefully chosen and tuned, for each application, by human experts. Also, some techniques of parameter identification, like the evaluation of the Degree of Support (DoS) in fuzzyTech, may disguise for structural identification. This approach is however computationally feasible for very small variable spaces (under 10 variables) because of the limitations of the initial rule set size in fuzzyTech. Better results are obtained in Matlab with the approach utilizing the neurofuzzy (ANFIS) learning together with the sequential forward search for variable identification.

The real-life water treatment case study was performed using commercial products Matlab and *fuzzyTech*. Both products are applicable in the sense that they produce executable and verifiable FIS models. Models produced with fuzzyTech may be automatically converted to RapidBase models. The Matlab output requires either additional conversion from the Takagi-Sugeno to Mamdani model or an extension to the RapidBase inference engine to support the Takagi-Sugeno model.

References

[BCGK99] P. P. Bonissone, Y-T Chen, K. Goebel and P. S. Khedkar. Hybrid Soft Computing Systems: Industrial and Commercial Applications. *Proceedings of the IEEE*, pp 1641--1667, vol. 87, no. 9, September 1999.

[BVH99] R. Babuska, H.B. Verbruggen, and H. Hellendoorn. Promising fuzzy modeling and control methodologies for industrial applications. In *Proc. of European Symposium on Intelligent Techniques ESIT'99*, AB-02, Crete, Greece, June 1999.

[ETG98] Mohammad R. Emami, I. Burhan Türksen, and Andrew A. Goldenberg. Development of A Systematic Methodology of Fuzzy Logic Modeling. *IEEE Transactions on Fuzzy Systems*, **6**(3): 346-361 (1998).

[HC00] Tzung-Pei Hong and Jyh-Bin Chen. Processing Individual Fuzzy Attributes for Fuzzy Rule Induction. *Fuzzy Sets and Systems* **112**(1): 127-140 (2000).

[HC99] Tzung-Pei Hong and Jyh-Bin Chen. Finding relevant attributes and membership functions. *Fuzzy Sets and Systems* **103**(3): 389-404 (1999).

[Hii00] Mikko Hiirsalmi. Prediction of quality indicators based on measurement history. VTT/TTE Research Report TTE1-2000-28.

[IRV97] J. Iivarinen, J. Rauhamaa and A. Visa. An Adaptive Two-Stage Approach to Classification of Surface Defects. In *Proc. of The 10th Scandinavian Conference on Image Analysis*, vol. I, pp. 317-322, Lappeenranta, Finland, June 9-11, 1997.

[Jan93] Jyh-Shing Roger Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man and Cybernetics*, **23** (3): 665-685 (1993).

[Kim+97] Euntai Kim, Minkee Park, Seunghwan Ji and Mignon Park. A New Approach to Fuzzy Modeling. *IEEE Transactions on Fuzzy Systems*, **5**(3): 328-337 (1997).

[Man74] E.H. Mamdani. Applications of fuzzy algorithms for simple dynamic plant. *Proc. IEE*, **121**(12), pp. 1585-1588 (1974).

[Mat00] Fuzzy logic Toolbox User's Guide, Version 2, The mathWorks Inc., (2000).

[MJ94] T. Munakata, Y. Jani. Fuzzy Systems: An Overview. *Communications of The ACM*, **7**(3), March 1994, pp. 69-76.

[Par+99] Minkee Park, Seunghwan Ji, Euntai Kim and Mignon Park. A new approach to the identification of a fuzzy model. *Fuzzy Sets and Systems* **104**(2), pp. 169-181 (1999).

[PW00] A. Pesonen, A. Wolski. Quantified and Temporal Fuzzy Reasoning for Active Monitoring in RapidBase. *Proc. of TOOLMET2000 Symposium – Tool Environments and Development Methods or Intelligent Systems*, Oulu, Finland, April 2000, 227-242.

[SVAH99] O. Simula, J. Vesanto, E. Alhoniemi and J. Hollmen. Analysis and Modeling Of Complex Systems Using the Self-Organizing Map. Chapter in *Neuro-Fuzzy Techniques for Intelligent Information Systems*, 1999.

[SY93] M. Sugeno and T. Yasukawa. A Fuzzy Logic Based Approach to Qualitative Modeling. *IEEE Transactions on Fuzzy Systems*, **1**(1): 7-31 (1993).

[ST83] M. Sugeno and T. Takagi. Multidimentional Fuzzy Reasoninng. *Fuzzy Sets and Systems*, **9**(2), 1983.

[TS85] T. Takagi and M. Sugeno. Fuzzy Identification of Systems and its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-15** (1), pp. 116-132 (1985).

[WM92] Li-Xin Wang and Jerry M. Mendel. Generating Fuzzy Rules by Learning from Examples. *IEEE Transactions on Systems, Man, and Cybernetics*, **22**(6), pp. 1414-1427 (1992).

[W94] L-X Wang. *Adaptive Fuzzy systems and Control: Design and Stability Analysis*. Prentice-Hall, Englewood Cliffs, NJ (1994).

[WB98] A. Wolski, and T. Bouaziz. Fuzzy Triggers: Incorporating Imprecise Reasoning into Active Databases. *Proc. 14th International Conference on*

Data Engineering (ICDE'98), Feb. 23-27, 1998, Orlando, Florida, pp. 108-155. IEEE Computer Society Press, 1998.

[WL95] L. Wang, and R. Langari. Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques. *IEEE Trans. on Fuzzy Systems*, 3(4): 454 -458 (Nov. 1995)

[XL87] Chen-Wei Xu and Ying-Zai Lu. Fuzzy Model Identification and Self-Learning for Dynamic Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-17** (4): 683-689 (1987).

[YF94] Ronald R. Yager and Dimitar P. Filev. *Essentials of Fuzzy Modeling and Control*. John Wiley & Sons (1994)

[Zad65] Lofti A. Zadeh. Fuzzy Sets. *Information Control*, **8**(3), June 1965, pp. 338-353.