

## A FUZZY SYSTEM WITH $\varepsilon$ -INSENSITIVE LEARNING OF PREMISES AND CONSEQUENCES OF IF-THEN RULES

JACEK M. ŁĘSKI\*, TOMASZ CZOGAŁA\*\*

\* Institute of Electronics, Silesian University of Technology  
ul. Akademicka 16, 44–100 Gliwice, Poland  
e-mail: jleski@polsl.pl

\*\* Institute of Medical Technology and Equipment  
ul. Roosevelta 118A, 41–800 Zabrze, Poland  
e-mail: tczogala@onet.pl

First, a fuzzy system based on if-then rules and with parametric consequences is recalled. Then, it is shown that the global and local  $\varepsilon$ -insensitive learning of the above fuzzy system may be presented as a combination of both an  $\varepsilon$ -insensitive gradient method and solving a system of linear inequalities. Examples are given of using the introduced method to design fuzzy models of real-life data. Simulation results show an improvement in the generalization ability of a fuzzy system trained by the new method compared with the traditional and other  $\varepsilon$ -insensitive learning methods.

**Keywords:** fuzzy system, generalization ability, global and local  $\varepsilon$ -insensitive learning, extraction of fuzzy if-then rules.

### 1. Introduction

The support vector machine (SVM) is historically the first method based on the main result of statistical learning theory, i.e., the generalization ability of a machine depends on both the empirical risk on a training set and the complexity of this machine. For an in-depth study of statistical learning theory, see (Vapnik, 1995; 1998; 1999). The SVM has been successfully applied to a wide variety of classification and regression problems.

In the last few years, there has been increasing interest in fuzzy systems which incorporate well-known tools from statistical learning theory. Fuzzy clustering with a weighted (or fuzzy)  $\varepsilon$ -insensitive loss function was introduced in (Łęski, 2001; 2003a; 2004a). The above method leads to improved robustness to outliers with respect to traditional fuzzy clustering methods. Support vector fuzzy regression machines were introduced in (Hong and Hwang, 2003). A support vector interval regression network was established in (Jeng *et al.*, 2003). A differentiable approximation of the misclassification rate and using the empirical risk minimization principle to improve the learning of a neuro-fuzzy classifier is proposed in (Castellano *et al.*, 2004). The work (Chiang and Hao, 2003) reports the support vector fuzzy clustering method. An  $\varepsilon$ -insensitive approach to the learning of neuro-fuzzy systems was introduced in (Łęski, 2001) and extended in (Łęski, 2002a; 2002b; 2003b; 2004b). A similar approach to training a classifier, called the fuzzy support vector ma-

chine, was independently introduced in (Lin and Wang, 2002). The concept of the fuzzy kernel perceptron is presented in (Chen *et al.*, 2002).

From the above-mentioned methods the  $\varepsilon$ -insensitive approach to the learning of neuro-fuzzy systems is of special interest in this work. This approach is based on the premise that human learning, as well as thinking, is tolerant of imprecision. Instead of the usually used quadratic loss function, an  $\varepsilon$ -insensitive loss function is used which assumes a zero loss for the difference between a model and the reality less than some pre-set value, noted as  $\varepsilon$ . If this difference is greater than  $\varepsilon$ , then the loss increases linearly.  $\varepsilon$ -insensitive learning is based on the connection between fuzzy modeling and statistical learning theory where easy control of system complexity is permitted. Learning tolerant to imprecision always leads to a better generalization ability and robustness to outliers compared with the traditional methods (Łęski, 2003b). In the previous works  $\varepsilon$ -insensitive learning was used for the consequences of if-then rules only. The premises of if-then rules were selected using preliminary fuzzy clustering in the input space. Such selected premises remain unchanged in the learning process. However, in the traditional approach to fuzzy (or neuro-fuzzy) modeling both premises and consequences of if-then rules are adjusted during the process of learning (Czogala and Łęski, 2000; Jang *et al.*, 1997; Rutkowska, 2001; Rutkowski and Cpalka, 2003). Thus, the main purpose of this work is to answer the following question:

Does the method for adjusting the premises of if-then rules based on the gradient descent approach lead to fuzzy systems with an improved generalization ability with respect to  $\varepsilon$ -insensitive learning used for the consequences of if-then rules only? A key to the approach in this paper are the following changes to the  $\varepsilon$ -insensitive learning of a fuzzy system:

- (1) An  $\varepsilon$ -insensitive gradient descent approach to adjusting the parameters of premises. Such an approach is common in the context of traditional neuro-fuzzy modeling.
- (2) An  $\varepsilon$ -insensitive learning method for the parameters of consequences by a modified Solving a System of Linear Inequalities algorithm. In this approach the problem of the estimation of consequence parameters with the control of the complexity of the model will be shown for both global and local approaches.
- (3) A fuzzy system based on if-then rules with parametric consequences. In this case, the fuzzy systems well-known from the literature may be treated as a special type of this fuzzy system.

The main goal of this work is to introduce global and local  $\varepsilon$ -insensitive learning of a fuzzy system, in which both premises and consequences are adjusted during learning. For the sake of generality, new learning methods will be shown on a fuzzy system based on if-then rules with parametric consequences. The next goal is to investigate the generalization ability of the fuzzy system obtained by means of new learning methods for real-world benchmark data. We also include a comparison with traditional approaches to fuzzy modeling and a state-of-the-art method based on the support vector machine.

The problem solved in this paper may be more formally defined as follows: Suppose we have the training set  $\mathcal{T}^{(N)} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$ , where  $N$  stands for data cardinality, and each independent input datum  $\mathbf{x}_i \in \mathbb{R}^t$  has a corresponding dependent output datum  $y_i \in \mathbb{R}$ . Let us define the testing set  $\mathcal{T}e^{(M)} = \{(\mathbf{x}_{N+1}, y_{N+1}), (\mathbf{x}_{N+2}, y_{N+2}), \dots, (\mathbf{x}_{N+M}, y_{N+M})\}$ , where  $M$  denotes data cardinality. We seek a knowledge-base of a fuzzy system on the basis of the training set  $\mathcal{T}^{(N)}$ . The quality of the obtained knowledge-base is measured using the generalization ability. It refers to producing a reasonable output of a fuzzy system for a data pair unused during the process of extracting the knowledge-base. Throughout this paper, the generalization error is determined as a root mean squared error (RMSE) calculated on the testing set  $\mathcal{T}e^{(M)}$ . Indeed, the error refers to the difference between a fuzzy model output and a desired (output) datum from the testing set.

The remaining part of the paper is structured as follows: A short description of a fuzzy system based on if-then rules with parametric consequences is recalled in Section 2. Section 3 presents an introduction of an  $\varepsilon$ -insensitive learning method of the parameters of conse-

quences by a modified Solving a System of Linear Inequalities algorithm ( $\varepsilon$ LSSLI). An  $\varepsilon$ -insensitive gradient descent approach to adjusting the parameters of premises is presented in Section 4. Hybrid learning algorithms of the parameters of premises and consequences are introduced in Section 5. Section 6 presents simulation results and a discussion of fuzzy modeling of real-world high-dimensional data. Finally, conclusions are drawn in Section 7.

We now describe our notation. All vectors and matrices will be denoted in boldface. Vectors will be in columns. Transposed vectors will be denoted by superscript  $\top$ . The notation  $\mathbf{X} \in \mathbb{R}^{m \times t}$ ,  $\mathbf{y} \in \mathbb{R}^r$  denotes a real  $m \times t$  matrix and a real  $r$ -dimensional vector, respectively. The  $\text{diag}(\mathbf{x})$  denotes a diagonal matrix with diagonal elements taken from the vector  $\mathbf{x}$ .  $\mathbf{0}_{N \times 1}$  and  $\mathbf{1}_{M \times 1}$  denote a vector of dimension  $N \times 1$  with all entries equal to 0 and a vector of dimension  $M \times 1$  with all entries equal to 1, respectively. The identity matrix will be denoted by  $\mathbf{I}$ .  $\star_T$  stands for  $t$ -norm  $T$  operation. For a fuzzy set  $A$ ,  $\mu_A(x)$  denotes its membership function. Throughout this paper, the notation  $\text{Area}(\mu_A(x))$  will denote the area under the membership function  $\mu_A(x)$ . The  $\varepsilon$ -insensitive loss function will be denoted by  $\lceil \xi \rceil_\varepsilon = \max(|\xi| - \varepsilon, 0)$ , where  $\varepsilon \geq 0$  is an insensitivity parameter and  $\xi$  is an arbitrary scalar.

## 2. Fuzzy Systems with Parametric Consequences in If-Then Rules

In this section, fuzzy rules with parametric consequences will be used to recall the important fuzzy systems which are basic in further deliberations. The above systems are selected because the fuzzy systems well-known from the literature may be treated as a special type of a fuzzy system based on if-then rules with parametric consequences.

Let us assume that  $I$  fuzzy if-then rules with  $t$  inputs and one output (MISO) are given. The  $i$ -th rule in which the consequent is represented by a fuzzy set  $B^{(i)}(\boldsymbol{\theta})$  whose membership function depends on parameter vector  $\boldsymbol{\theta}$  may be written in the following form (Czogała and Łęski, 2000):

$$\begin{aligned} \mathbb{R}^{(i)} : & \text{IF } X_1 \text{ IS } A_1^{(i)} \text{ AND } \dots \text{ AND } X_t \text{ IS } A_t^{(i)}, \\ & \text{THEN } Y \text{ IS } B^{(i)}(\boldsymbol{\theta}) \end{aligned} \quad (1)$$

or in a pseudo-vector notation:

$$\mathbb{R}^{(i)} : \text{IF } \mathbf{X} \text{ IS } \mathbf{A}^{(i)}, \text{ THEN } Y \text{ IS } B^{(i)}(\boldsymbol{\theta}), \quad (2)$$

where

$$\mathbf{X} = [X_1, X_2, \dots, X_t]^\top \quad (3)$$

and  $X_1, X_2, \dots, X_t$  and  $Y$  are linguistic variables which may be interpreted as inputs of a fuzzy system and the output of that system.  $A_1^{(i)}, A_2^{(i)}, \dots, A_t^{(i)}$  are linguistic values of the linguistic variables  $X_1, X_2, \dots, X_t$  and  $B^{(i)}(\theta)$  is a linguistic value of the linguistic variable  $Y$ . The vector  $\theta$  consists of parameters of input fuzzy sets (their height or the localization of their centers of gravity). Fuzzy modeling allows finding nonlinear models of reality where knowledge is obtained as a set of the above-mentioned if-then rules with linguistically interpreted propositions. Fuzzy modeling is based on the premise that human thinking is tolerant of imprecision and the real world is too complicated to be described precisely (Zadeh, 1973). Fuzzy modeling has an intrinsic inconsistency. It may perform thinking tolerant of imprecision, but the traditional learning methods are zero-tolerant of imprecision. The approach to fuzzy modeling presented in this paper is based on the premise that human learning, as well as thinking, is tolerant of imprecision.

A collection of the above-written rules for  $i = 1, 2, \dots, I$  creates a rule base which may be fired by the singleton inputs

$$X_1 \text{ IS } x_{01} \text{ AND } \dots \text{ AND } X_t \text{ IS } x_{0t} \quad (4)$$

or, shortly,

$$\mathbf{X} \text{ IS } \mathbf{x}_0. \quad (5)$$

In the above case the  $i$ -th fuzzy if-then rule has the form

$$\begin{aligned} \mathbb{R}^{(i)} : & \text{ IF } x_{01} \text{ IS } A_1^{(i)} \text{ AND } \dots \text{ AND } x_{0t} \text{ IS } A_t^{(i)}, \\ & \text{ THEN } Y \text{ IS } B^{(i)}(\mathbf{x}_0) \end{aligned} \quad (6)$$

and may be called the fuzzy if-then rule with a moving fuzzy consequent (Łęski and Czogała, 1999). In the case when fuzzy singletons are used as inputs, the vector  $\theta$  may consist of localizations of singletons only. So,  $\theta = \mathbf{x}_0$ . A conclusion output fuzzy set for the  $i$ -th rule may be written in the form (Czogała and Łęski, 2000):

$$\mu_{B^{(i)}}(y, \mathbf{x}_0) = \mathcal{F}_i(\mathbf{x}_0) \star_{T_i} \mu_{B^{(i)}}(y, \mathbf{x}_0), \quad (7)$$

where  $\star_{T_i}$  stands for a conjunctive interpretation of the if-then rule and

$$\begin{aligned} \mathcal{F}_i(\mathbf{x}_0) &= \mu_{A_1^{(i)}}(x_{01}) \star_T \dots \star_T \mu_{A_t^{(i)}}(x_{0t}) \\ &= \mu_{\mathbf{A}^{(i)}}(\mathbf{x}_0) \end{aligned} \quad (8)$$

denotes the firing strength of the  $i$ -th rule. Equation (8) represents an explicit connective (AND) of the predicates  $x_{0k} \text{ IS } A_k^{(i)}$ ;  $k = 1, 2, \dots, t$ , in the premise of the  $i$ -th fuzzy if-then rule.

A final crisp value of the system output for a normalized sum as aggregation, COG defuzzification and the

algebraic product used for if-then rules interpretation may be evaluated from the formula (Czogała and Łęski, 2000):

$$y_0(\mathbf{x}_0) = \frac{\sum_{i=1}^I y^{(i)}(\mathbf{x}_0) \text{Area}(\mu_{B^{(i)}}(y, \mathbf{x}_0))}{\sum_{k=1}^I \text{Area}(\mu_{B^{(i)}}(y, \mathbf{x}_0))}, \quad (9)$$

where  $B^{(i)}$  is a resulting conclusion for the  $i$ -th if-then rule before aggregation and  $y^{(i)}$  is the location of the center of gravity of the fuzzy set  $B^{(i)}$ :

$$\begin{aligned} y^{(i)}(\mathbf{x}_0) &= \text{COG}(\mu_{B^{(i)}}(y, \mathbf{x}_0)) \\ &= \frac{\int y \mu_{B^{(i)}}(y, \mathbf{x}_0) dy}{\int \mu_{B^{(i)}}(y, \mathbf{x}_0) dy}. \end{aligned} \quad (10)$$

Usually, we assume that locations of fuzzy sets in consequences are linear combinations of the inputs  $\mathbf{x}_0$ :

$$y^{(i)}(\mathbf{x}_0) = p_0^{(i)} + \tilde{\mathbf{p}}^{(i)\top} \mathbf{x}_0 = \mathbf{p}^{(i)\top} \mathbf{x}'_0, \quad (11)$$

where  $\mathbf{p}^{(i)} = [p_0^{(i)}, \tilde{\mathbf{p}}^{(i)\top}]^\top$ ,  $\tilde{\mathbf{p}}$  denotes the parameter vector with a bias element excluded, the superscript  $\top$  stands for transposition and  $\mathbf{x}'_0$  denotes an extended input vector  $\mathbf{x}'_0 = [1, \mathbf{x}_0^\top]^\top$ .

It is worth noting that for singletons as fuzzy sets in consequences of if-then rules and the algebraic product used for the interpretation of if-then rules, the well-known Takagi-Sugeno-Kang fuzzy system is obtained (Czogała and Łęski, 2000; 2001). On the other hand, the Mamdani-Assilan fuzzy system is obtained for fuzzy sets in consequences which do not depend on  $\mathbf{x}_0$  (Czogała and Łęski, 2000; 2001).

Let us assume that the consequents  $B^{(i)}$  of the  $i$ -th if-then rule have symmetric triangle (isosceles triangle) membership functions with the width of the triangle base equal to  $w^{(i)}$ . The area  $(\mu_{B^{(i)}}(y))$  should be determined to evaluate the fuzzy system output on the basis of (9). From the definition of  $B^{(i)}$  and (7) we have

$$\text{Area}(\mu_{B^{(i)}}(y, \mathbf{x}_0)) = \frac{w^{(i)}}{2} \mathcal{F}_i(\mathbf{x}_0). \quad (12)$$

A crisp value of the output for the fuzzy system can be evaluated from the following formula (Czogała and Łęski, 2000):

$$y_0(\mathbf{x}_0) = \frac{\sum_{i=1}^I w^{(i)} \mathcal{F}_i(\mathbf{x}_0) \mathbf{p}^{(i)\top} \mathbf{x}'_0}{\sum_{k=1}^I w^{(k)} \mathcal{F}_k(\mathbf{x}_0)}. \quad (13)$$

For the sake of simplicity  $w^{(i)} = \text{const}$ , for  $i = 1, 2, \dots, I$ , is used in this work. Thus, a crisp value of the output may be written as

$$y_0(\mathbf{x}_0) = \frac{\sum_{i=1}^I \mathcal{F}_i(\mathbf{x}_0) \mathbf{p}^{(i)\top} \mathbf{x}'_0}{\sum_{k=1}^I \mathcal{F}_k(\mathbf{x}_0)}. \quad (14)$$

The system described by (14) can be interpreted as a mixture of expert models. The response of the  $i$ -th expert (if-then rule) for the input  $\mathbf{x}_0$  is  $\mathbf{p}^{(i)\top} \mathbf{x}'_0$ . The final output of the system is obtained as a weighted average of local experts outputs. The non-negative weight of association between  $\mathbf{x}_0$  and the  $i$ -th expert is  $\mathcal{F}_i(\mathbf{x}_0)$ . Thus, nonlinear models of reality can be described as a combination of simple linear models. It is well-known in machine learning that too precise learning on a training set leads to overfitting (overtraining), which results in a poor generalization ability. The generalization ability is interpreted as the production of a reasonable decision for data previously unseen in the training process. Statistical learning theory has recently emerged as a general theory for the estimation of dependencies from a finite set of data (Vapnik, 1995). The most important issue in this theory is the Structural Risk Minimization (SRM) induction principle. The SRM principle suggests a tradeoff between the quality of an approximation and the complexity of the approximating function (Vapnik, 1995; 1998). A measure of the approximation function complexity (or capacity) is called the VC-dimension. One of the simplest methods to control the VC-dimension is to change the insensitivity parameter  $\varepsilon$  in the loss function and to enforce the flatness of the approximating function by a regularization constant (Vapnik, 1995; 1998; 1999). The above approach will be used in the fuzzy system described by (14) in the next section.

The output value  $y_0$  of the system in Eqn. (14) may be considered to be a linear combination of unknown parameters  $\mathbf{p}^{(i)}$ . If we introduce the following notation:

$$S^{(i)}(\mathbf{x}_0) = \frac{\mathcal{F}_i(\mathbf{x}_0)}{\sum_{k=1}^I \mathcal{F}_k(\mathbf{x}_0)}, \quad (15)$$

$$\mathbf{d}(\mathbf{x}_0) = \left[ S^{(1)}(\mathbf{x}_0) \mathbf{x}'_0{}^\top, S^{(2)}(\mathbf{x}_0) \mathbf{x}'_0{}^\top, \dots, S^{(I)}(\mathbf{x}_0) \mathbf{x}'_0{}^\top \right]^\top, \quad (16)$$

$$\mathbf{P} = \left[ \mathbf{p}^{(1)\top}, \mathbf{p}^{(2)\top}, \dots, \mathbf{p}^{(I)\top} \right]^\top, \quad (17)$$

then (14) may be written in the form

$$y_0(\mathbf{x}_0) = \mathbf{d}(\mathbf{x}_0)^\top \mathbf{P}. \quad (18)$$

Indeed, membership functions for the premises of if-then rules (6) should be selected to evaluate (18). Typically, the problem of estimation values of these functions is solved by means of preliminary clustering of the input part of the training set using the fuzzy  $c$ -means method (Chen *et al.*, 1998; Pedrycz, 1984; Czogała and Łęski, 2000 and Setnes, 2000). The premise fuzzy set of the  $i$ -th rule has a membership function  $\mu_{\mathbf{A}^{(i)}}(\mathbf{x}_0) : \mathbb{R}^t \rightarrow [0, 1]$ . In case of Gaussian membership functions and the algebraic product used as the  $t$ -norm modeling AND, the fuzzy premise is defined as

$$\begin{aligned} \mu_{\mathbf{A}^{(i)}}(\mathbf{x}_0) &\triangleq \prod_{j=1}^t \exp \left[ -\frac{\left(x_{0j} - c_j^{(i)}\right)^2}{2s_j^{(i)}} \right] \\ &= \exp \left[ -\frac{1}{2} \sum_{j=1}^t \frac{\left(x_{0j} - c_j^{(i)}\right)^2}{s_j^{(i)}} \right], \quad (19) \end{aligned}$$

where the parameters  $c_j^{(i)}, s_j^{(i)}, i = 1, 2, \dots, c; j = 1, 2, \dots, t$  are centers and dispersions of membership functions for the  $i$ -th rule and the  $j$ -th input variable, respectively.

Usually, the learning of fuzzy systems presented above may be executed using the following schemes (Czogała and Łęski, 2000; Jang *et al.*, 1997 and Rutkowska, 2001):

- the parameters from the premises and consequents of if-then rules are adjusted separately. First, the premise parameters are adjusted using unsupervised learning—the clustering of the input data using the fuzzy  $c$ -means method. Second, the consequents parameters are adjusted by means of the gradient descent method or the least squares method.
- the parameters are adjusted in two-phase learning. First, as in the previous method, the premise parameters are adjusted using unsupervised learning. Second, all parameters (premise and consequent) are adjusted by means of the gradient descent method.
- First, the premise parameters are adjusted using unsupervised learning. Finally, in each iteration the parameters  $\mathbf{p}^{(i)}$  are estimated on the basis of the least squares method, whereas the parameters  $c_j^{(i)}, s_j^{(i)}$  by means of the gradient descent method.

There are two approaches to using the least squares method: first, to solve one global Least Squares (LS) problem, for all if-then rules; second, to solve  $I$  independent weighted LS problems, one for each if-then rule (Łęski, 2002a). The first approach leads to better global

performance, while the second one leads to more reliable local performance. In this work both approaches, i.e., global learning and local learning will be used to introduce the idea of  $\varepsilon$ -insensitive learning of premises and consequences of a fuzzy system.

### 3. $\varepsilon$ -insensitive Learning of Consequences of If-Then Rules

LS learning methods use the quadratic loss function to match the reality and a fuzzy model. In this case only perfect matching between the reality and the model leads to a zero loss. The approach to fuzzy modeling presented in this section is based on the premise that human learning as well as thinking is tolerant of imprecision. Hence, an  $\varepsilon$ -insensitive loss function is used and learning methods based on this loss function lead to so-called  $\varepsilon$ -insensitive fuzzy modeling (Łeński, 2001; 2002a; 2003b; 2004a). In further parts of this section, the problem of the estimation of consequence parameters with the control of the complexity of the model will be shown for both global and local approaches. We seek the consequence parameters vector  $\mathbf{P}$  on the basis of a set of independent identically distributed (i.i.d.) data pairs called a training set  $\mathcal{T}^{(N)} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ , where  $N$  is data cardinality and each independent datum  $\mathbf{x}_i \in \mathbb{R}^t$  has a corresponding dependent datum  $y_i \in \mathbb{R}$ .

#### 3.1. Problem Formulation

Using the  $\varepsilon$ -insensitive loss function for an arbitrary scalar  $\xi$  (Vapnik, 1995):

$$|\xi|_{\varepsilon} = \begin{cases} 0, & |\xi| \leq \varepsilon, \\ |\xi| - \varepsilon, & |\xi| > \varepsilon, \end{cases} \quad (20)$$

where  $\varepsilon \geq 0$  denotes the insensitivity parameter, the global learning criterion function has the following form (Łeński, 2003b):

$$\min_{\mathbf{P} \in \mathbb{R}^{I(t+1)}} I_g(\mathbf{P}) \triangleq \sum_{n=1}^N |y_n - \mathbf{P}^T \mathbf{d}(\mathbf{x}_n)|_{\varepsilon} + \frac{\tau}{2} \tilde{\mathbf{P}}^T \tilde{\mathbf{P}}, \quad (21)$$

where  $\tilde{\mathbf{P}} = [\tilde{\mathbf{p}}^{(1)T}, \tilde{\mathbf{p}}^{(2)T}, \dots, \tilde{\mathbf{p}}^{(I)T}]^T$  is a narrowed vector  $\mathbf{P}$ , with excluded components corresponding to the biases (see Eqn. (11));  $(\mathbf{x}_n, y_n)$  is the  $n$ -th pair from the training set. The second term in (21) is related to the minimization of the Vapnik-Chervonenkis dimension (complexity) of the regression model (Vapnik, 1998). The parameter  $\tau > 0$  controls the trade-off between regression model complexity and the amount up to which errors are tolerated.

In the local approach to learning fuzzy system parameters of each if-then rule are obtained separately.

Thus, to obtain  $I$  if-then rules the minimization of the respective criterion function should be done  $I$ -times. Using the  $\varepsilon$ -insensitive loss function, the local learning criterion function for the  $i$ -th if-then rule has the form (Łeński, 2001; 2002a):

$$\min_{\mathbf{p}^{(i)} \in \mathbb{R}^{t+1}} I_l^{(i)}(\mathbf{p}^{(i)}) = \sum_{n=1}^N S^{(i)}(\mathbf{x}_n) |y_n - \mathbf{p}^{(i)T} \mathbf{x}_n'|_{\varepsilon} + \frac{\tau}{2} \tilde{\mathbf{p}}^{(i)T} \tilde{\mathbf{p}}^{(i)}. \quad (22)$$

The above equation is called the weighted  $\varepsilon$ -insensitive estimator (or fuzzy  $\varepsilon$ -insensitive) with complexity control (Łeński, 2002a).

As we saw in (Łeński, 2001; 2002a; 2002b), the determination of the parameters  $\mathbf{P}$  or  $\mathbf{p}^{(i)}$  leads to a quadratic programming (QP) problem with bound constraints and one linear equality constraint. For a large training set, standard optimization techniques quickly become intractable in their memory and time requirements. Thus, the work (Łeński, 2004a) proposes a computationally effective algorithm called incremental learning. An alternative approach called the iterative QP solution, which determines the parameters in the fuzzy modeling problem, is presented in (Łeński, 2002a). The third approach to  $\varepsilon$ -insensitive learning presented in (Łeński, 2003b) leads to a problem of solving a system of linear inequalities. In this paper this idea is used to solve the problem of fuzzy modeling with  $\varepsilon$ -insensitive learning. However, a new modified method of solving a system of inequalities will be introduced.

If we define  $\mathbf{X}_d \triangleq [\mathbf{d}(\mathbf{x}_1), \mathbf{d}(\mathbf{x}_2), \dots, \mathbf{d}(\mathbf{x}_N)]^T \in \mathbb{R}^{N \times I(t+1)}$ ,  $\mathbf{y} = [y_1, y_2, \dots, y_N]^T \in \mathbb{R}^N$ , the minimization problem (21) can be rewritten in the matrix form

$$\min_{\mathbf{P}} I_g(\mathbf{P}) = |\mathbf{y} - \mathbf{X}_d \mathbf{P}|_{\mathbf{1}_{N \times 1}, \varepsilon} + \frac{\tau}{2} \mathbf{P}^T \tilde{\mathbf{I}} \mathbf{P}, \quad (23)$$

where  $\mathbf{1}_{N \times 1}$  denotes the vector of dimension  $N \times 1$  with all entries equal to 1;  $\tilde{\mathbf{I}} = \text{diag}(\underbrace{[0, \mathbf{1}_{t \times 1}^T, 0, \mathbf{1}_{t \times 1}^T, \dots, 0, \mathbf{1}_{t \times 1}^T]^T}_{I\text{-times}})$ ;  $|\cdot|_{\mathbf{h}, \varepsilon}$  de-

notes a weighted Vapnik loss function defined for the vector argument  $\mathbf{e} = [e_1, e_2, \dots, e_N]^T$  and weights  $\mathbf{h} = [h_1, h_2, \dots, h_N]^T$  as (Łeński, 2002a):

$$|\mathbf{e}|_{\mathbf{h}, \varepsilon} \triangleq \sum_{n=1}^N h_n |e_n|_{\varepsilon}. \quad (24)$$

If we denote  $\mathbf{X}_1 \triangleq [\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_N]^T \in \mathbb{R}^{N \times (t+1)}$ , then the minimization problem (22) for the  $i$ -th rule may

Table 1. Notations used in global and local  $\varepsilon$ -insensitive learning.

Notation	Global learning	Local learning ( $i$ -th rule)
$v$	$I(t + 1)$	$t + 1$
$\mathbf{r}$	$\mathbf{P} = [\mathbf{p}^{(1)\top}, \mathbf{p}^{(2)\top}, \dots, \mathbf{p}^{(I)\top}]^\top$	$\mathbf{p}^{(i)} = [p_0^{(i)}, \tilde{\mathbf{p}}^{(i)\top}]^\top$
$\mathbf{Z}$	$\mathbf{X}_d \triangleq [\mathbf{d}(\mathbf{x}'_1), \mathbf{d}(\mathbf{x}'_2), \dots, \mathbf{d}(\mathbf{x}'_N)]^\top$	$\mathbf{X}_1 \triangleq [\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_N]^\top$
$\mathbf{g}$	$\mathbf{1}_{N \times 1}$	$\mathbf{h}^{(i)} = [S^{(i)}(\mathbf{x}_1), S^{(i)}(\mathbf{x}_2), \dots, S^{(i)}(\mathbf{x}_N)]^\top$
$\tilde{\mathbf{K}}$	$\tilde{\mathbf{I}} = \text{diag} \left( \underbrace{[0, \mathbf{1}_{t \times 1}^\top, 0, \mathbf{1}_{t \times 1}^\top, \dots, 0, \mathbf{1}_{t \times 1}^\top]}_{I\text{-times}} \right)^\top$	$\tilde{\mathbf{J}} = \text{diag} \left( [0, \mathbf{1}_{t \times 1}^\top]^\top \right)$

be easily rewritten in the form similar to (23), i.e.,

$$\min_{\mathbf{p}^{(i)}} I_l^{(i)}(\mathbf{p}^{(i)}) = \|\mathbf{y} - \mathbf{X}_1 \mathbf{p}^{(i)}\|_{\mathbf{h}^{(i)}, \varepsilon} + \frac{\tau}{2} \mathbf{p}^{(i)\top} \tilde{\mathbf{J}} \mathbf{p}^{(i)}, \quad (25)$$

where

$$\tilde{\mathbf{J}} = \text{diag} \left( [0, \mathbf{1}_{t \times 1}^\top]^\top \right);$$

$$\mathbf{h}^{(i)} = [S^{(i)}(\mathbf{x}_1), S^{(i)}(\mathbf{x}_2), \dots, S^{(i)}(\mathbf{x}_N)]^\top.$$

Thus, both global and local learning of consequence parameters can be presented as a solution of the following minimization problem:

$$\min_{\mathbf{r}} I(\mathbf{r}) = \|\mathbf{y} - \mathbf{Zr}\|_{\mathbf{g}, \varepsilon} + \frac{\tau}{2} \mathbf{r}^\top \tilde{\mathbf{K}} \mathbf{r}, \quad (26)$$

where  $\mathbf{r} = [r_1, r_2, \dots, r_v]^\top \in \mathbb{R}^v$ ,  $\mathbf{Z} \in \mathbb{R}^{N \times v}$ ,  $\mathbf{g} = [g_1, g_2, \dots, g_N]^\top \in \mathbb{R}^N$ ,  $\tilde{\mathbf{K}} \in \mathbb{R}^{v \times v}$ . The meaning of the notations  $\mathbf{r}$ ,  $\mathbf{Z}$ ,  $\tilde{\mathbf{K}}$  and others is summarized in Table 1.

Now, the problem of  $\varepsilon$ -insensitive learning of the consequences of rules for both global and local approaches is equivalent to the minimization of the criterion (26). An iterative method for the minimization of (26) was proposed in (Łęski, 2002b; 2003b). A modified Solving a System of Linear Inequalities algorithm ( $\varepsilon$ LSSLI) will be introduced in the next subsection.

### 3.2. A New Algorithm for Learning Rule Consequences

The main problem with the minimization of (26) is how to make the first term mathematically tractable. We see that the first term is equal to zero when the following requirements are satisfied:  $\mathbf{Zr} + \varepsilon \mathbf{1}_{N \times 1} > \mathbf{y}$  and  $\mathbf{Zr} - \varepsilon \mathbf{1}_{N \times 1} < \mathbf{y}$ . These inequalities may be rewritten in the form  $\mathbf{Zr} - \mathbf{y} + \varepsilon \mathbf{1}_{N \times 1} > \mathbf{0}_{N \times 1}$  and  $-\mathbf{Zr} +$

$\mathbf{y} + \varepsilon \mathbf{1}_{N \times 1} < \mathbf{0}_{N \times 1}$ , where  $\mathbf{0}_{N \times 1}$  denotes the vector of dimension  $N \times 1$  with all entries equal to zero. We may multiply both sides of the above inequalities by the parameter  $a > 0$ :  $a\mathbf{Zr} - a(\mathbf{y} + \varepsilon \mathbf{1}_{N \times 1}) > \mathbf{0}_{N \times 1}$  and  $-a\mathbf{Zr} + a(\mathbf{y} + \varepsilon \mathbf{1}_{N \times 1}) < \mathbf{0}_{N \times 1}$ . The role of this parameter will be explained later.

Let us now define the following vectors and matrix:  $\mathbf{r}_a \triangleq a\mathbf{r}$ ,  $\mathbf{r}_e \triangleq [\mathbf{r}_a^\top, a]^\top$ ,

$$\mathbf{Z}_e \triangleq \begin{bmatrix} \mathbf{Z} & -\mathbf{y} + \varepsilon \mathbf{1}_{N \times 1} \\ -\mathbf{Z} & \mathbf{y} + \varepsilon \mathbf{1}_{N \times 1} \end{bmatrix}. \quad (27)$$

Taking into account the above definitions, our inequalities may be written as

$$\mathbf{Z}_e \mathbf{r}_e > \mathbf{0}_{2N \times 1}. \quad (28)$$

Thus, if the above system of inequalities is fulfilled, then the first term of (26) is equal to zero. In practically interesting cases, not all inequalities in the above system may be fulfilled (except for the case where  $\varepsilon$  is large enough to make all the data fall into the insensitivity region). Note that the solution of the inequality system  $\mathbf{Z}_e \mathbf{r}_e > \mathbf{0}_{2N \times 1}$  is equivalent to the solution of the equality system  $\mathbf{Z}_e \mathbf{r}_e = \mathbf{b}$ , where  $\mathbf{b}$  is an arbitrary positive vector,  $\mathbf{b} > \mathbf{0}_{2N \times 1}$ . Indeed, we do not know  $\mathbf{b}$ . However, the method described below enables us to obtain both  $\mathbf{r}_e$  and  $\mathbf{b}$  in such a way as to maximize the degree of fulfillment of (28).

Let us first define the error vector as  $\mathbf{e} = \mathbf{Z}_e \mathbf{r}_e - \mathbf{b}$ . If the  $i$ -th,  $1 \leq i \leq 2N$ , component of  $\mathbf{e}$  is greater than zero, i.e.,  $e_i > 0$ , then the  $i$ -th equality is not fulfilled. However, the corresponding  $i$ -th inequality is fulfilled. Additionally, we can get  $e_i = 0$  by increasing the respective component of  $\mathbf{b}$ , i.e.,  $b_i$ . In this case, the requirement  $\mathbf{b} > \mathbf{0}_{2N \times 1}$  will still be satisfied. On the other hand, if the  $i$ -th component of  $\mathbf{e}$  is negative, then neither the  $i$ -th equality nor the  $i$ -th inequality are fulfilled. Note that it is impossible to decrease  $b_i$  and fulfill the condition  $b_i > 0$ . In the original algorithm (Łęski, 2002b), the value

of  $b_i$  remains unchanged. However, in the current formulation of the problem we have the parameter  $a > 0$  (in the original algorithm  $a \equiv 1$ ). The advantage of this formulation is the possibility to decrease the components of  $\mathbf{b}$  and simultaneously fulfill the condition  $\mathbf{b} > \mathbf{0}_{2N \times 1}$ . For example, if we set  $a = 0.5$ , then both sides of  $\mathbf{Z}_e \mathbf{r}_e = \mathbf{b}$  are divided by 2. So, the elements of the vector  $\mathbf{b}$  are divided by 2. Although all components are divided simultaneously, the respective components can be increased in successive iterations. In conclusion, we can iteratively increase or divide the components of the vector  $\mathbf{b}$  until the components of the error vector  $\mathbf{e}$  are negative for inequalities which are not fulfilled and equal to zero for inequalities which are fulfilled.

Our minimization problem (26) can be approximated by the following:

$$\min_{\mathbf{r}_e; \mathbf{b} > \mathbf{0}_{2N \times 1}} I(\mathbf{r}_e, \mathbf{b}) \triangleq (\mathbf{Z}_e \mathbf{r}_e - \mathbf{b})^\top \mathbf{G}_e (\mathbf{Z}_e \mathbf{r}_e - \mathbf{b}) + \frac{\tau}{2} \mathbf{r}_e^\top \tilde{\mathbf{L}} \mathbf{r}_e, \quad (29)$$

where

$$\mathbf{G}_e = \text{diag} \left( [\mathbf{g}^\top, \mathbf{g}^\top]^\top \right)$$

and

$$\tilde{\mathbf{L}} = \begin{bmatrix} \tilde{\mathbf{K}} & 0 \\ 0 & 0 \end{bmatrix}. \quad (30)$$

As we can see, the first term of (29) may be rewritten as  $\mathbf{e}^\top \mathbf{G}_e \mathbf{e}$ . From the above we know that the elements of the vector  $\mathbf{b}$  will be iteratively changed until the elements of the error vector  $\mathbf{e}$  are equal to zero for satisfied inequalities and negative for unsatisfied inequalities. So, the first term of (29) is a measure of the degree of fulfillment of the analysed inequality system. For mathematical simplicity, the above criterion is an approximation of (26) because the squared error is used rather than the absolute error. After describing the algorithm for the squared error, in the further part of this section, an algorithm for the absolute error will be obtained using an iteratively re-weighted least-squares method.

We obtain conditions for optimality by differentiating (29) with respect to  $\mathbf{r}_e, \mathbf{b}$ :

$$\begin{cases} \frac{\partial I(\mathbf{r}_e, \mathbf{b})}{\partial \mathbf{r}_e} = 2\mathbf{Z}_e^\top \mathbf{G}_e (\mathbf{Z}_e \mathbf{r}_e - \mathbf{b}) + \tau \tilde{\mathbf{L}} \mathbf{r}_e, \\ \frac{\partial I(\mathbf{r}_e, \mathbf{b})}{\partial \mathbf{b}} = -2\mathbf{G}_e (\mathbf{Z}_e \mathbf{r}_e - \mathbf{b}) \end{cases} \quad (31)$$

and setting the results equal to zero:

$$\begin{cases} \mathbf{r}_e = \left( \mathbf{Z}_e^\top \mathbf{G}_e \mathbf{Z}_e + \frac{\tau}{2} \tilde{\mathbf{L}} \right)^{-1} \mathbf{Z}_e^\top \mathbf{G}_e \mathbf{b}, \\ \mathbf{e} = \mathbf{Z}_e \mathbf{r}_e - \mathbf{b} = \mathbf{0}. \end{cases} \quad (32)$$

From the first equation (32) we see that vector  $\mathbf{r}_e$  depends on the vector  $\mathbf{b}$ . The only way to prevent  $\mathbf{b}$  from converging to zero is to start with  $\mathbf{b} > \mathbf{0}_{2N \times 1}$  and to refuse to decrease any of its components. Ho and Kashyap proposed an iterative algorithm for alternately determining  $\mathbf{r}_e$  and  $\mathbf{b}$ , where the components of  $\mathbf{b}$  cannot decrease (Ho and Kashyap, 1965; 1966). The solution is obtained in an iterative way. The vector  $\mathbf{r}_e$  is determined on the basis of the first equation from (32), i.e.,  $\mathbf{r}_e^{[k]} = \left( \mathbf{Z}_e^\top \mathbf{G}_e \mathbf{Z}_e + \frac{\tau}{2} \tilde{\mathbf{L}} \right)^{-1} \mathbf{Z}_e^\top \mathbf{G}_e \mathbf{b}^{[k]}$ , where the superscript  $[k]$  denotes the iteration index. The components of the vector  $\mathbf{b}$  are modified by the components of the error vector  $\mathbf{e}$ , but only in the case when it results in increasing the components of  $\mathbf{b}$ . Otherwise, the components of  $\mathbf{b}$  remain unmodified. So, we write this modification as follows:

$$\mathbf{b}^{[k+1]} = \mathbf{b}^{[k]} + \rho \mathbf{G}_e (\mathbf{e}^{[k]} + |\mathbf{e}^{[k]}|), \quad (33)$$

where  $\rho > 0$  is a parameter. Taking into account the second equation (31), we see that (33) may be treated as the gradient descent modification of  $\mathbf{b}$ :

$$\mathbf{b}^{[k+1]} = \mathbf{b}^{[k]} - \frac{\rho}{2} \left( \left. \frac{\partial I(\mathbf{r}_e, \mathbf{b})}{\partial \mathbf{b}} \right|_{\mathbf{b}=\mathbf{b}^{[k]}} - \left. \frac{\partial I(\mathbf{r}_e, \mathbf{b})}{\partial \mathbf{b}} \right|_{\mathbf{b}=\mathbf{b}^{[k]}} \right). \quad (34)$$

In the above equation the traditional gradient descent method is modified in such a way that if any component of the gradient vector is positive, then it is set to zero. So, we see from (33) that the components of  $\mathbf{b}$  can increase the only. However, the introduction of the parameter  $a$ , as explained before, causes all components of  $\mathbf{b}$  to be divided by  $a > 0$ .

For mathematical simplicity, the criterion (29) is an approximation of (26) because the squared error rather than the absolute error is used. Note that the criterion (29) may be rewritten for the absolute error as

$$\begin{aligned} I(\mathbf{r}_e, \mathbf{b}) &= [\mathbf{g}^\top, \mathbf{g}^\top] |\mathbf{Z}_e \mathbf{r}_e - \mathbf{b}| + \frac{\tau}{2} \mathbf{r}_e^\top \tilde{\mathbf{L}} \mathbf{r}_e \\ &= [\mathbf{g}^\top, \mathbf{g}^\top] |\mathbf{e}| + \frac{\tau}{2} \mathbf{r}_e^\top \tilde{\mathbf{L}} \mathbf{r}_e. \end{aligned} \quad (35)$$

Taking into account the equality

$$\begin{aligned} |\mathbf{e}| &= \sum_{i=1}^{2N} |e_i| = \sum_{i=1}^{2N} \frac{e_i^2}{|e_i|} \\ &= \mathbf{e}^\top \text{diag} \left( \frac{1}{|e_1|}, \frac{1}{|e_2|}, \dots, \frac{1}{|e_{2N}|} \right) \mathbf{e}, \end{aligned} \quad (36)$$

where  $e_i$  is the  $i$ -th component of the error vector, the

criterion (35) may be rewritten as

$$I(\mathbf{r}_e, \mathbf{b}) = \mathbf{e}^\top \text{diag} \left( \frac{g_1}{|e_1|}, \frac{g_2}{|e_2|}, \dots, \frac{g_N}{|e_{2N}|} \right) \mathbf{e} + \frac{\tau}{2} \mathbf{r}_e^\top \tilde{\mathbf{L}} \mathbf{r}_e. \quad (37)$$

Thus, comparing (29) and (37), the absolute error criterion equivalent to (26) is easily obtained by selecting the following diagonal weight matrix:  $\mathbf{G}_e = \text{diag}(g_1/|e_1|, \dots, g_N/|e_N|, g_1/|e_{N+1}|, \dots, g_N/|e_{2N}|)$ . However, the error vector depends on  $\mathbf{r}_e$ , so, we use the vector  $\mathbf{r}_e$  from the previous iteration. This procedure is based on the premise that the sequential vectors  $\mathbf{r}_e^{[k]}$  differ imperceptibly near the optimum solution.

The procedure of seeking optimal  $\mathbf{r}_e$  and  $\mathbf{b}$  may be summarized in the following steps:

1. Fix  $\tau \geq 0$ ,  $\rho > 0$  and  $\mathbf{G}_e^{[1]} = \text{diag}([\mathbf{g}^\top, \mathbf{g}^\top]^\top)$ . Initialize  $\mathbf{b}^{[1]} > \mathbf{0}_{2N \times 1}$ . Set the iteration index  $k = 1$ ,
2.  $\mathbf{r}_e^{[k]} = (\mathbf{Z}_e^\top \mathbf{G}_e^{[k]} \mathbf{Z}_e + \frac{\tau}{2} \tilde{\mathbf{L}})^{-1} \mathbf{Z}_e^\top \mathbf{G}_e^{[k]} \mathbf{b}^{[k]}$ ,
3.  $\mathbf{e}^{[k]} = \mathbf{Z}_e \mathbf{r}_e^{[k]} - \mathbf{b}^{[k]}$ ,
4.  $\mathbf{G}_e^{[k+1]} = \text{diag}(g_1/|e_1^{[k]}|, \dots, g_N/|e_N^{[k]}|, g_1/|e_{N+1}^{[k]}|, \dots, g_N/|e_{2N}^{[k]}|)$ ,
5.  $\mathbf{b}^{[k+1]} = \mathbf{b}^{[k]} + \rho \mathbf{G}_e^{[1]} (\mathbf{e}^{[k]} + |\mathbf{e}^{[k]}|)$ ,
6. if  $k > 1$  and  $\left\| \mathbf{r}_e^{[k]} / a^{[k]} - \mathbf{r}_e^{[k-1]} / a^{[k-1]} \right\|_2 < \kappa$ , then go to Step 7, else  $k = k + 1$ , go to Step 2.
7.  $\mathbf{r} = \mathbf{r}_a^{[k]} / a^{[k]}$ . STOP.

An iterative method for the minimization of (26) based on  $\varepsilon$ -insensitive Learning by Solving a System of Linear Inequalities ( $\varepsilon$ LSSLI) was proposed in (Łęski, 2002b; 2003b). The above-introduced algorithm is a modified version of this algorithm and may be called  $\varepsilon$ LSSLI1.

#### 4. $\varepsilon$ -insensitive Learning of Rule Premises

Let us observe that for Gaussian membership functions of the rule premises (19), the following unknown parameters should be determined:  $c_j^{(i)}$ ,  $s_j^{(i)}$  for  $j = 1, 2, \dots, t$  and  $i = 1, 2, \dots, I$ . Usually, the above-mentioned unknown parameters are estimated by means of fuzzy  $c$ -means clustering (Łęski, 2002a). Indeed, in our case we have  $I$  clusters. So, the name fuzzy  $I$ -means method will be better. In this method each input vector  $\mathbf{x}_n$ ;  $n = 1, 2, \dots, N$ , is assigned to clusters represented by the prototypes  $\mathbf{v}_i$ ;  $i = 1, \dots, I$  measured by the grade of membership  $u_{in} \in [0, 1]$ . The  $(I \times N)$ -dimensional partition matrix

$\mathbf{U}$  comes from the set of all possible fuzzy partitions into  $I$  clusters and is defined by

$$\mathfrak{S}_{fI} = \left\{ \mathbf{U} \in \mathbb{R}^{I \times N} \mid \forall_{\substack{1 \leq i \leq I \\ 1 \leq n \leq N}} u_{in} \in [0, 1], \sum_{i=1}^I u_{in} = 1, 0 < \sum_{n=1}^N u_{in} < N \right\}. \quad (38)$$

The fuzzy  $I$ -means criterion function has the form (Bezdek, 1982):

$$J_m(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^I \sum_{n=1}^N (u_{in})^m d_{in}^2, \quad (39)$$

where  $\mathbf{U} \in \mathfrak{S}_{fI}$ ,  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_I] \in \mathbb{R}^{t \times I}$  and  $m$  is a weighting exponent in  $(1, \infty)$ . The quantity  $d_{in}^2$  is the following norm:  $d_{in}^2 = \|\mathbf{x}_n - \mathbf{v}_i\|^2 = (\mathbf{x}_n - \mathbf{v}_i)^\top (\mathbf{x}_n - \mathbf{v}_i)$ .

It can be proved that a local minimum of the criterion (39) may be obtained by an iterative method of commutative modification of the partition matrix and prototypes (Bezdek, 1982):

$$\forall_{\substack{1 \leq i \leq I \\ 1 \leq n \leq N}} u_{in} = \left[ \sum_{j=1}^I \left( \frac{d_{in}}{d_{jn}} \right)^{\frac{2}{m-1}} \right]^{-1}, \quad (40)$$

$$\forall_{1 \leq i \leq I} \mathbf{v}_i = \frac{\sum_{n=1}^N (u_{in})^m \mathbf{x}_n}{\sum_{n=1}^N (u_{in})^m}. \quad (41)$$

The optimal partition is a fixed point of (40) and (41), and the solution is obtained from the Picard iteration. The fuzzy  $I$ -means can be described in the following steps:

1. Fix  $I$  ( $1 < I < N$ ),  $m \in (1, \infty)$ . Initialize  $\mathbf{U}^{(0)} \in \mathfrak{S}_{fI}$ . Set the iteration index,  $j = 0$ .
2. Calculate centers for the  $j$ -th iteration  $\mathbf{V}^{(j)} = [\mathbf{v}_1^{(j)}, \mathbf{v}_2^{(j)}, \dots, \mathbf{v}_I^{(j)}]$  using (41) and  $\mathbf{U}^{(j)}$ .
3. Update the fuzzy partition matrix  $\mathbf{U}^{(j+1)}$  for the  $(j+1)$ -th iteration using (40).
4. If  $\|\mathbf{U}^{(j+1)} - \mathbf{U}^{(j)}\|_F > \xi_{\mathbf{U}}$ , then  $j = j + 1$  and go to Step 2, else STOP.

$\|\cdot\|_F$  denotes the Frobenius norm ( $\|\mathbf{U}\|_F^2 = \text{Tr}(\mathbf{U}\mathbf{U}^\top) = \sum_i \sum_n u_{in}^2$ ) and  $\xi_{\mathbf{U}}$  is a pre-set parameter. There is no theoretical basis for the optimal selection of  $m$ , and usually  $m = 2$  is chosen.

According to the above-written algorithm, the calculations are initialized using a random partition matrix



U which fulfils conditions from (38). Such a method leads to the local minimum of the criterion (39). Therefore, the most frequently used solution consists in multiple repeated calculations in accordance with the above algorithm for various random realizations of the initial partition matrix. Usually, validity indices which measure the cluster quality are used. One of the most popular validity indices is the extended Xie-Beni index (Xie and Beni, 1991):

$$\mathcal{V}_{XB} = \frac{\sum_{n=1}^N \sum_{i=1}^I (u_{in})^m d_{in}^2}{N \min_{k \neq i} \|\mathbf{v}_k - \mathbf{v}_i\|}. \quad (42)$$

Indeed, we search a fuzzy  $I$ -partition for which the index  $\mathcal{V}_{XB}$  is minimal, that is, minimizing the compactness of clusters whilst maximizing their separation. As a result of preliminary clustering of the training set, the following assumption for the initialization of the premises of parameters is made:

$$c_j^{(i)} = \frac{\sum_{n=1}^N u_{in} x_{nj}}{\sum_{n=1}^N u_{in}} \quad (43)$$

and

$$(s_j^{(i)})^2 = \frac{\sum_{n=1}^N u_{in} (x_{nj} - c_j^{(i)})^2}{\sum_{n=1}^N u_{in}}, \quad (44)$$

where  $u_{in}$  denotes the membership degree of the vector  $\mathbf{x}_n = [x_{n1}, x_{n2}, \dots, x_{nt}]^T$  from the training set to the  $i$ -th cluster (to the premise of the  $i$ -th if-then rule).

Frequently, the above-described method of obtaining the premises of rules is used for the initialization of a gradient descent method (Łęski and Czogała, 1999). The measure of the error of the system output value may be defined for a single pair from the training set as

$$E_n = L(y_n - y_0(\mathbf{x}_n)), \quad (45)$$

where  $y_n, y_0(\mathbf{x}_n)$  denote the desired (target) and actual value of the system output for  $\mathbf{x}_n$ , respectively. The function  $L(\cdot)$  stands for a loss function. Most frequently a quadratic loss function is used, that is,  $L(\cdot) = \frac{1}{2}(\cdot)^2$ . The  $\varepsilon$ -insensitive loss function (20) will be used in this paper. For the entire training set, we define the error function as the average of  $E_n$ :

$$E = \frac{1}{N} \sum_{n=1}^N E_n. \quad (46)$$

In the so-called batch mode of learning, parameters are updated after the presentation of all examples from

the training set, called an epoch. Thus, the minimization of the error  $E$  is made iteratively (for the parameter  $\theta \in \{c_j^{(i)}, s_j^{(i)}\}_{i,j=1}^{j=I, i=I}$ ):

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \frac{\partial E}{\partial \theta} \Big|_{\theta=\theta_{\text{old}}}, \quad (47)$$

where  $\eta > 0$  is the learning rate parameter.

In the sequential mode of learning (the stochastic mode), parameters are updated after the presentation of each example from the training set. From the point of view of real-mode the sequential mode is preferred. In addition, given that examples are presented in a random manner to the system, the search is stochastic. In this case, it is less probable for a learning algorithm to be trapped in a local minimum.

Taking into account (14) we may express the partial derivatives of the error  $E_n$  with respect to the unknown parameters from the premises of rules as

$$\frac{\partial E_n}{\partial c_j^{(i)}} = \mathcal{A}_n \frac{[y^{(i)}(\mathbf{x}_n) - y_0(\mathbf{x}_n)] \mathcal{F}_i(\mathbf{x}_n) x_{nj} - c_j^{(i)}}{\sum_{k=1}^I \mathcal{F}_k(\mathbf{x}_n) (s_j^{(i)})^2}, \quad (48)$$

$$\frac{\partial E_n}{\partial s_j^{(i)}} = \mathcal{A}_n \frac{[y^{(i)}(\mathbf{x}_n) - y_0(\mathbf{x}_n)] \mathcal{F}_i(\mathbf{x}_n) (x_{nj} - c_j^{(i)})^2}{\sum_{k=1}^I \mathcal{F}_k(\mathbf{x}_n) (s_j^{(i)})^3}, \quad (49)$$

where

$$\mathcal{A}_n = \frac{\partial E_n}{\partial y_0(\mathbf{x}_0)} \Big|_{\mathbf{x}_0=\mathbf{x}_n}. \quad (50)$$

Indeed, for the quadratic loss function, we obtain  $\mathcal{A}_n = -(y_n - y_0(\mathbf{x}_n))$ . Using the  $\varepsilon$ -insensitive loss function, the measure of the error for the  $n$ -th example has the form (Vapnik, 1998):

$$E_n = |y_n - y_0(\mathbf{x}_n)|_{\varepsilon} \triangleq \begin{cases} 0, & |y_n - y_0(\mathbf{x}_n)| \leq \varepsilon, \\ |y_n - y_0(\mathbf{x}_n)| - \varepsilon, & |y_n - y_0(\mathbf{x}_n)| > \varepsilon. \end{cases} \quad (51)$$

In the above case, the quantity  $\mathcal{A}_n$  takes the form

$$\mathcal{A}_n = \frac{\partial E_n}{\partial y_0(\mathbf{x}_0)} \Big|_{\mathbf{x}_0=\mathbf{x}_n} \quad (52)$$

$$= \begin{cases} 0, & |y_n - y_0(\mathbf{x}_n)| \leq \varepsilon, \\ \text{sgn}(y_n - y_0(\mathbf{x}_n)), & |y_n - y_0(\mathbf{x}_n)| > \varepsilon, \end{cases} \quad (53)$$

where  $\text{sgn}(\cdot)$  denotes the signum function.

A very simple operation speeding up convergence is proposed by Jang *et al.* (1997). In (47), the learning rate parameter is selected in a special way:

$$\theta_{\text{new}} = \theta_{\text{old}} - \frac{v}{\sqrt{\sum_{i=1}^r \left(\frac{\partial E}{\partial \theta_i}\right)^2_{\theta_i=(\theta_i)_{\text{old}}}}} \frac{\partial E}{\partial \theta} \Big|_{\theta=\theta_{\text{old}}}, \quad (54)$$

where  $r$  denotes the number of optimized parameters, in our case,  $r = 2tI$ ;  $v > 0$  is the so-called step size. If in four successive steps of gradient descent learning the error  $E$  increases and decreases commutatively, then the step size is decreased, that is, multiplied by  $n_D < 1$ . However, if in four successive steps of gradient descent learning the error  $E$  decreases, then the step size is increased, that is, multiplied by  $n_I > 1$ .

### 5. Hybrid Learning Algorithms

Learning algorithms which incorporate techniques described in previous sections will be introduced in this section. Let us assume that the following parameters are given: the number of if-then rules  $I \geq 2$ , the insensitivity parameter  $\varepsilon \geq 0$  and the regularization parameter  $\tau > 0$ . A method of obtaining these parameters will be described later. The following learning algorithms are proposed:

- **$\varepsilon$ -LS- $\varepsilon$ -gradient.** First, the parameters of the premises of rules are obtained using the fuzzy  $I$ -means algorithm. Next, the parameters of the consequents of rules are obtained using  $\varepsilon$ LSSL1 (in a local or global manner). Then, the above initial parameters of rules are adjusted iteratively. Each iteration consists of  $\varepsilon$ -insensitive gradient descent modification of the parameters of premises and determining the parameters of consequences by the  $\varepsilon$ LSSL1 method (local or global). For the gradient method the parameters of consequences are treated as fixed, and for the  $\varepsilon$ LSSL1 method the parameters of premises are treated as fixed. The iterations are stopped when parameters in successive iterations differ imperceptibly.
- **$\varepsilon$ -LS-gradient.** This algorithm is similar to the  $\varepsilon$ -LS- $\varepsilon$ -gradient method, however, the quadratic loss function is used in the gradient descent modification of the parameters of premises.
- **$\varepsilon$ -LS.** First, the parameters of the premises of rules are obtained using the fuzzy  $I$ -means algorithm. Then, the parameters of the consequents of rules are obtained using the  $\varepsilon$ LSSL1 method (local or global). The iterative modification of the parameters of rules is not performed.

- **LS-gradient.** First, the parameters of the premises of rules are obtained using the fuzzy  $I$ -means algorithm. Next, the parameters of the consequents of rules are obtained using a weighted Least Squares (LS) method (in a local or global manner). Then, the above initial parameters of rules are adjusted iteratively. Each iteration consists of the gradient descent modification (the quadratic loss function) of the parameters of premises and determining the parameters of consequences by the LS method (local or global). For the gradient method the parameters of consequences are treated as fixed, and for the LS method the parameters of premises are treated as fixed. The iterations are stopped when parameters in successive iterations differ imperceptibly.
- **LS.** First, the parameters of the premises of rules are obtained using the fuzzy  $I$ -means algorithm. Then, the parameters of the consequents of rules are obtained using the LS method (local or global). The iterative modification of the parameters of rules is not performed.

Indeed, the above-mentioned algorithms are simplified versions of the  $\varepsilon$ -LS- $\varepsilon$ -gradient method. Thus, let us start from the presentation of the  $\varepsilon$ -LS- $\varepsilon$ -gradient method. The initial values of premise parameters are obtained using the fuzzy  $I$ -means algorithm. For a fixed number of clusters,  $I \geq 2$ , the clustering is repeated 25 times for a different random initialization of the partition matrix  $\mathbf{U}$ . The clustering algorithm is presented at the beginning of the previous section. The following values of parameters are used:  $m = 2$  (the weighting exponent) and  $\xi_{\mathbf{U}} = 1 \cdot 10^{-3}$  (the parameter in the stopping condition). Finally, clusters corresponding to the minimal value of the Xie-Beni index (42) are used. Then, the initial values of the parameters of the premises  $c_j^{(i)}, s_j^{(i)}$  for  $j = 1, 2, \dots, t$  and  $i = 1, 2, \dots, I$  are determined using (43) and (44).

The further part of the algorithm consists of commutative performance of  $\varepsilon$ LSSL1 and  $\varepsilon$ -insensitive gradient descent modification of the parameters of premises. The  $\varepsilon$ LSSL1 algorithm was precisely described at the end of Section 3.2. The meaning of the notation used in this algorithm for local and global learning is summarized in Table 1. The algorithm is performed for given values of  $\varepsilon$  (the insensitivity parameter) and  $\tau$  (the regularization parameter). The iterations are stopped as soon as the norm in a successive pair of  $\mathbf{r}_e$  vectors is less than  $\kappa = 1 \cdot 10^{-3}$ . The selection of the parameter values  $\mathbf{b}^{[1]} > \mathbf{0}_{2N \times 1}$  and  $\rho > 0$  will be done in the experimental part of the paper. Indeed, for local learning, the  $\varepsilon$ LSSL1 algorithm is performed  $I$  times, one for each rule and for global learning simultaneously for all rules. The  $\varepsilon$ -insensitive gradient descent method modifies the parameters of premises,

i.e.,  $c_j^{(i)}, s_j^{(i)}$  for  $j = 1, 2, \dots, t$  and  $i = 1, 2, \dots, I$  in the batch mode using (54). The gradients of  $E$  with respect to  $c_j^{(i)}, s_j^{(i)}$  are cumulated by means of (48), (49) and (53). Initially, the step size is set to 0.01. If in four successive steps of gradient descent learning the error  $E$  increases and decreases commutatively, then the step size is decreased, i.e.,  $v \leftarrow 0.9 v$ . On the other side, if in four successive steps of gradient descent learning the error  $E$  decreases, then the step size is increased, that is,  $v \leftarrow 1.1 v$ .

Now, the whole  $\varepsilon$ -LS- $\varepsilon$ -gradient algorithm may be summarized in the following steps:

1. Fix  $\varepsilon \geq 0, \tau > 0, I \geq 2$ .
2. Repeat 25 times the fuzzy  $I$ -means algorithm for a different random initialization of the partition matrix  $\mathbf{U}$ .
3. Initial values of the parameters of the premises  $c_j^{(i),[1]}, s_j^{(i),[1]}$  are determined using (43) and (44) from clustering corresponding to the minimal value of the Xie-Beni index (42).
4. Set the iteration index,  $\ell = 1$ , and the step size,  $v = 0.01$ .
5. Determine the parameters of the consequences  $w^{(i),[\ell]}$  using the  $\varepsilon$ LSSL1 algorithm with  $c_j^{(i),[\ell]}, s_j^{(i),[\ell]}$ .
6. Cumulate gradients with the respect parameters  $c_j^{(i),[\ell]}, s_j^{(i),[\ell]}$ .
7. Update the premise parameters  $c_j^{(i),[\ell+1]}, s_j^{(i),[\ell+1]}$  using (54).
8. Determine the error measure (46) for the  $\ell$ -th iteration,  $E^{[\ell]}$ .
9. If  $\ell > 4$  and  $E^{[\ell]} < E^{[\ell-1]}$  and  $E^{[\ell-1]} < E^{[\ell-2]}$  and  $E^{[\ell-2]} < E^{[\ell-3]}$  and  $E^{[\ell-3]} < E^{[\ell-4]}$ , then  $v \leftarrow 1.1 v$ .
10. If  $\ell > 4$  and  $E^{[\ell]} < E^{[\ell-1]}$  and  $E^{[\ell-1]} > E^{[\ell-2]}$  and  $E^{[\ell-2]} < E^{[\ell-3]}$  and  $E^{[\ell-3]} > E^{[\ell-4]}$ , then  $v \leftarrow 0.9 v$ .
11. If  $\ell > 1$  and  $|E^{[\ell]} - E^{[\ell-1]}| < 1 \cdot 10^{-4}$ , then stop, else  $\ell \leftarrow \ell + 1$  go to Step 5.

In the above algorithm the superscript  $[\ell]$  denotes the iteration index. If in the  $\varepsilon$ -LS- $\varepsilon$ -gradient algorithm  $\mathcal{A}_n = -(y_n - y_0(\mathbf{x}_n))$  will be used instead of the quantity  $\mathcal{A}_n$  from (53), then the  $\varepsilon$ -LS- $\varepsilon$ -gradient algorithm is obtained. In the  $\varepsilon$ -LS algorithm the iterative modification of rule parameters is not performed. Thus this algorithm consists of the following steps:

1. Fix  $\varepsilon \geq 0, \tau > 0, I \geq 2$ .
2. Repeat 25 times the fuzzy  $I$ -means algorithm for a different random initialization of the partition matrix  $\mathbf{U}$ .
3. Values of the parameters of the premises  $c_j^{(i)}, s_j^{(i)}$  are determined using (43) and (44) from clustering corresponding to the minimal value of the Xie-Beni index (42).
4. Determine the parameters of the consequences  $w^{(i)}$  using the  $\varepsilon$ LSSL1 algorithm with  $c_j^{(i)}, s_j^{(i)}$ . STOP.

The LS-gradient algorithm may be easily obtained by replacing the determination of the parameters of consequences by the least squares (LS) method in the  $\varepsilon$ -LS- $\varepsilon$ -gradient algorithm. Using notations from the previous sections, the global LS solution to the consequent parameters can be written in the matrix form as

$$\mathbf{P} = (\mathbf{X}_d^\top \mathbf{X}_d)^{-1} \mathbf{X}_d^\top \mathbf{y}, \quad (55)$$

where  $\mathbf{X}_d \triangleq [\mathbf{d}(\mathbf{x}'_1), \mathbf{d}(\mathbf{x}'_2), \dots, \mathbf{d}(\mathbf{x}'_N)]^\top \in \mathbb{R}^{N \times I(t+1)}$ ,  $\mathbf{y} = [y_1, y_2, \dots, y_N]^\top \in \mathbb{R}^N$ . The local LS solution to the consequent parameters of the  $i$ -th rule can be written in the matrix form as

$$\mathbf{p}^{(i)} = (\mathbf{X}_1^\top \mathbf{G}^{(i)} \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{G}^{(i)} \mathbf{y}, \quad (56)$$

where  $\mathbf{X}_1 \triangleq [\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_N]^\top \in \mathbb{R}^{N \times (t+1)}$  and  $\mathbf{G}^{(i)} = \text{diag}([S^{(i)}(\mathbf{x}_1), S^{(i)}(\mathbf{x}_2), \dots, S^{(i)}(\mathbf{x}_N)]^\top)$ . The same values of premise parameter as in the  $\varepsilon$ -insensitive learning algorithms are used.

Finally, the LS algorithm consists of the following steps:

1. Fix  $I \geq 2$ .
2. Repeat 25 times the fuzzy  $I$ -means algorithm for a different random initialization of the partition matrix  $\mathbf{U}$ .
3. Values of the parameters of the premises  $c_j^{(i)}, s_j^{(i)}$  are determined using (43) and (44) from clustering corresponding to the minimal value of the Xie-Beni index (42).
4. Determine the parameters of the consequences  $w^{(i)}$  using (55) and (56) with  $c_j^{(i)}, s_j^{(i)}$ . STOP.

## 6. Numerical Experiments and Discussion

Experiments were run on a Pentium IV 2.53 GHz processor running Windows XP and the MATLAB 7.0 environment. In all experiments,  $\mathbf{b}^{[1]} = \mathbf{1}_{2N \times 1}$  and  $\rho = 0.9$  were used in  $\varepsilon$ LSSLI and  $\varepsilon$ LSSLI1 methods. The iterations for the  $\varepsilon$ LSSLI method were stopped as soon as the Euclidean norm in a successive pair of the  $\mathbf{b}$  vectors was less than  $\kappa = 10^{-4}$ . The iterations for the  $\varepsilon$ LSSLI1 method were stopped as soon as the Euclidean norm in a successive pair of the  $\mathbf{r}_e/a$  vectors was less than  $\kappa = 10^{-4}$ . For global and local  $\varepsilon$ -insensitive learning the standard fuzzy  $c$ -means clustering method was used with the weighted exponent  $m = 2$ . The iterations were stopped as soon as the Frobenius norm in a successive pair of partition matrices was less than  $10^{-3}$ . For the number of clusters fixed the clustering was repeated 25 times for a different random initialization of the partition matrix. Finally, clusters corresponding to the minimal value of the Xie-Beni index were used.

### 6.1. Tests for the $\varepsilon$ LSSLI1 Method

The purpose of this experiment was to compare the performance of the  $\varepsilon$ LSSLI1 method proposed in the paper with that of the original method  $\varepsilon$ LSSLI. A two-dimensional (one input and one output) data set consists of a pair of true linear function with mixed Gaussian and Bernoulli-Gaussian random noise. The true but unknown (for algorithms) model is  $y = 1.5x + 4.5 + e$ , where  $e$  represents random noise. Thus, the true parameters are  $p_0 = 4.5$ ,  $p_1 = 1.5$ . The training set consists of 100 samples of the linear model. Each datum pair  $(x_n, y_n)$  was generated by the following technique: first, a uniform random number  $x_n$  was generated in  $(0, 100)$ , next, the value of  $y_n$  was obtained using the linear model and mixed Gaussian and Bernoulli-Gaussian random noise  $e = s + \mu$ . Bernoulli-Gaussian noise was generated as follows (Mendel, 1983):  $\mu_n = r_n q_n$ . In this model  $q$  has a Bernoulli distribution with the parameter  $\lambda$ :

$$\text{Prob}(q_n) = \begin{cases} \lambda, & q_n = 1, \\ 1 - \lambda, & q_n = 0. \end{cases} \quad (57)$$

The  $r$  and  $s$  have zero-mean Gaussian distributions with the variance  $\sigma_r^2, \sigma_s^2$ , respectively. The variables  $r, q$  and  $s$  are statistically independent. In the experiment, the following values of parameters were used:  $\lambda = 0.4$  and  $\sigma_r = 35$  and  $\sigma_s = 5$ . Parameter  $\tau$  was taken from the range 0 to 10 (the step 0.1) and  $\varepsilon$  was changed in the range from 0 to 5.0 (the step 0.1). The vector of weights was  $\mathbf{h}^{(i)} = \mathbf{1}_{N \times 1}$ . After the training stage using LS,  $\varepsilon$ LSSLI and  $\varepsilon$ LSSLI1 methods, the performance of these methods was determined as a Sum of Squared Error

(SSE) (difference between true and estimated model parameters):

$$\text{SSE} = (p_0 - \hat{p}_0)^2 + (p_1 - \hat{p}_1)^2, \quad (58)$$

where  $p_0, p_1$  denote true but unknown (for the algorithm) values of model parameters, and  $\hat{p}_0, \hat{p}_1$  stand for the parameters of the model obtained by the LS,  $\varepsilon$ LSSLI or  $\varepsilon$ LSSLI1 method. The above-mentioned training stage was repeated 100 times on different realizations of a training set. The average performance of the tested methods is as follows: LS—0.5690,  $\varepsilon$ LSSLI—0.2129 for  $\varepsilon = 0.1$  and  $\tau = 0.5$ ,  $\varepsilon$ LSSLI1—0.01101 for  $\varepsilon = 0.3$  and  $\tau = 2.6$ . It is worth noting that the mean number of iterations needed to fulfil the stop condition is 426 for  $\varepsilon$ LSSLI and 54 for the  $\varepsilon$ LSSLI1 method. Thus, experiments show that the proposed approach outperforms LS and performs competitively to the original  $\varepsilon$ LSSLI method.

### 6.2. Real World High-Dimensional Data

The purpose of these experiments was to compare the generalization ability of a fuzzy system learned using hybrid algorithms introduced in Section 5 and the classical (zero-tolerance) learning. The following benchmark databases were used:

- Data originating from the Box and Jenkins (Box and Jenkins, 1976) work concerning the identification of a gas oven. Air and methane were delivered into the gas oven (gas flow in ft/min—an input signal  $x(n)$ ) to obtain a mixture of gases containing CO<sub>2</sub> (percentage content—output signal  $y(n)$ ). The data consisted of 296 pairs of input-output samples with a 9 sec. sampling period. In order to identify the model, the following vectors were used as the input:  $\mathbf{x}_n \triangleq [y(n-1), \dots, y(n-4), x(n-1), \dots, x(n-6)]^T$  and the output:  $y_n \triangleq y(n)$ . The learning set consists of the first 100 pairs of data and the testing set consists of the remaining 190 pairs of data.
- ECG signal from the MIT-BIH database—the record numbered 100. The sampling frequency of that signal is equal to 360 Hz and the quantization step size is  $5\mu\text{V}$ . The learning process was conducted for the first 450 samples. The testing set consists of 5000 samples. The order of the model was equal to 4 and a nonlinear one-step predictor was built. Thus, the following vectors were used as the input:  $\mathbf{x}_n \triangleq [x(n-1), x(n-2), x(n-3), x(n-4)]^T$  and the output  $y_n \triangleq x(n)$ .

#### Remarks

1. It should be noted that for the Box-Jenkins dataset we used only 100 data points in an 11-dimensional

space to determine a fuzzy model. It is well known from statistics that as a function of space dimensionality, we need exponentially many data points to sample this space properly. However, in the light of statistical learning theory, the generalization ability is influenced by the complexity of the model rather than by the dimensionality of its input space. Thus, a model generalizes well when in a high-dimensional feature space it is chosen from a simple class of functions — for example, a linear class (the locations of fuzzy sets in consequences are linear combinations of the inputs  $x_0$ , cf.(11)).

2. For both of the above-described datasets, the training set is a small part of the available data samples. Thus, in this case a very pessimistic estimation of the generalization ability is determined, but, hopefully,  $\varepsilon$ -insensitive learning will be a good tool for constructing a fuzzy model even for a small number of data samples.
3. In most real-world problems a model should be constructed on the basis of a small given dataset. The ECG database is a good example. For Holter recordings a reasonable model of a signal should be designed using limited amount of information—usually the first few seconds of the signal. Such a model is used for on-line compression of the remaining part of the signal—usually 24 or 48 hours.

In all experiments the parameter  $\tau$  was taken from the set  $\{0.001, 0.01, 0.05, 0.1, 0.5\}$  and  $\varepsilon$  was changed in the range from 0 to 0.5 (the step 0.05). The number of if-then rules was changed from 2 to 6. After the training stage using the training part of data, the generalization ability of the designed model was determined as a root mean squared error (RMSE) on the test set. The training stage was repeated for each combination of the above values of parameters. Tables 2 and 4 show the RMSE for LS algorithms (global and local) obtained for Box-Jenkins and ECG databases, respectively. Tables 3 and 5

Table 2. RMSE obtained on the testing part of Box-Jenkins data by both local and global 0-insensitive learning (LS and LS-gradient methods).

I	LS learning		LS-gradient learning	
	local approach	global approach	local approach	global approach
2	0.3562	0.3629	0.3562	0.3426
3	0.3726	0.3854	0.3648	0.3588
4	0.4025	0.4417	0.3943	0.4416
5	0.4007	0.5662	0.3922	0.5406
6	0.4397	0.6686	0.3685	0.5716

Table 3. RMSE obtained on the testing part of Box-Jenkins data by both local and global  $\varepsilon$ -insensitive learning ( $\varepsilon$ -LS,  $\varepsilon$ -LS-gradient and  $\varepsilon$ -LS- $\varepsilon$ -gradient methods).

I	$\varepsilon$ -LS					
	local approach			global approach		
	RMSE	$\varepsilon$	$\tau$	RMSE	$\varepsilon$	$\tau$
2	0.3327	0.4	0.001	0.3313	0.45	0.05
3	0.3593	0.25	0.5	0.3516	0.15	0.05
4	0.3654	0.2	0.1	0.4273	0.25	0.05
5	0.4243	0.5	0.001	0.5134	0.15	0.01
6	0.4382	0.15	0.5	0.4977	0.4	0.001
I	$\varepsilon$ -LS-gradient					
	RMSE	$\varepsilon$	$\tau$	RMSE	$\varepsilon$	$\tau$
	2	0.3256	0.1	0.01	0.3162	0.2
3	0.3325	0.5	0.001	0.3287	0.45	0.001
4	0.3252	0.1	0.05	0.3305	0.35	0.01
5	0.3386	0.2	0.05	0.3206	0.3	0.01
6	0.3410	0.1	0.5	0.3258	0.1	0.1
I	$\varepsilon$ -LS- $\varepsilon$ -gradient					
	RMSE	$\varepsilon$	$\tau$	RMSE	$\varepsilon$	$\tau$
	2	0.3313	0.3	0.05	0.3222	0.15
3	0.3429	0.15	0.001	0.3249	0.15	0.001
4	0.3374	0.2	0.01	0.3235	0.15	0.5
5	0.3284	0.15	0.01	0.3323	0.05	0.05
6	0.3474	0.1	0.05	0.3355	0.2	0.5

show the lowest the RMSE for  $\varepsilon$ -insensitive learning algorithms (global and local) obtained for Box-Jenkins and ECG databases, respectively. The values of the parameters  $\varepsilon$  and  $\tau$  for which the lowest RMSE was obtained are also shown in these tables.

Table 4. RMSE obtained on the testing part of the ECG signal by both local and global 0-insensitive learning (LS and LS-gradient methods).

I	LS learning		LS-gradient learning	
	local approach	global approach	local approach	global approach
2	0.03171	0.03138	0.02877	0.02885
3	0.02820	0.02793	0.02819	0.02791
4	0.02761	0.02620	0.02760	0.02619
5	0.02437	0.03390	0.02437	0.02880
6	0.02573	0.02213	0.02573	0.02180

If we take these tables into account, several observations can be made. First of all, it should be noted that despite the number of if-then rules, learning tolerant of imprecision leads to better generalization comparing to zero-tolerant learning, for both databases. The best generaliza-

Table 5. RMSE obtained on the testing part of the ECG signal by both local and global  $\epsilon$ -insensitive learning ( $\epsilon$ -LS,  $\epsilon$ -LS-gradient and  $\epsilon$ -LS- $\epsilon$ -gradient).

I	$\epsilon$ -LS					
	local approach			global approach		
	RMSE	$\epsilon$	$\tau$	RMSE	$\epsilon$	$\tau$
2	0.02756	0.55	0.001	0.02944	0.45	0.01
3	0.02917	0.4	0.001	0.02971	0.55	0.001
4	0.02877	0.55	0.001	0.02746	0.1	0.1
5	0.02081	0.2	0.1	0.02013	0.55	0.05
6	0.02384	0.3	0.001	0.02394	0.55	0.01
$\epsilon$ -LS-gradient						
2	0.02681	0.1	0.1	0.02655	0.3	0.1
3	0.02556	0.05	0.5	0.02243	0.45	0.01
4	0.02488	0.2	0.001	0.02228	0.05	0.01
5	0.02081	0.15	0.1	0.01937	0.20	0.1
6	0.02175	0.25	0.05	0.02056	0.25	0.001
$\epsilon$ -LS- $\epsilon$ -gradient						
2	0.02723	0.05	0.01	0.02562	0.3	0.1
3	0.02755	0.2	0.001	0.02416	0.2	0.1
4	0.02698	0.05	0.001	0.02302	0.1	0.05
5	0.02081	0.15	0.1	0.02013	0.5	0.05
6	0.02175	0.35	0.001	0.02053	0.25	0.001

tion for each number of rules is obtained for the parameters  $\epsilon$  and  $\tau$  with a value different from zero. It must also be noted that we observe an improvement in the generalization ability for algorithms with gradient modification of premise parameters, i.e., the LS-gradient algorithm is better than the LS algorithm, and the  $\epsilon$ -LS-gradient algorithm is better than the  $\epsilon$ -LS algorithm. It is also important that the  $\epsilon$ -LS-gradient algorithm should slightly outperform the  $\epsilon$ -LS- $\epsilon$ -gradient algorithm. Thus, the best generalization ability is obtained by the  $\epsilon$ -LS-gradient learning algorithm.

The best generalization ability for Box-Jenkins data is obtained using the  $\epsilon$ -LS-gradient algorithm for global learning with  $c = 2$ ,  $\epsilon = 0.2$  and  $\tau = 0.001$ —RMSE = 0.3162. For the ECG database, the best generalization is obtained for the global  $\epsilon$ -LS-gradient algorithm with  $c = 5$ ,  $\epsilon = 0.2$  and  $\tau = 0.1$ —RMSE = 0.01937. Figure 1 shows the output of Box-Jenkins data as well as the output of the model obtained using the  $\epsilon$ -LS-gradient algorithm for global learning with  $c = 2$ ,  $\epsilon = 0.2$  and  $\tau = 0.001$ —upper; the course of the error signal for the training (left) and testing (right) part of data—lower. Figure 2 shows simulation results for the ECG signal. In these figures signals obtained by models are denoted by solid lines with point markers and the original signals are denoted by

solid lines. If we take these figures into account, it should be noted that models obtained by the  $\epsilon$ -LS-gradient algorithm on the basis of a limited amount of information generalize well. Errors in testing parts are slightly bigger than errors in training parts. For the Box-Jenkins dataset, many results obtained using all 296 data samples and reported in the literature are worse. For example, the following results may be enumerated: RMSE = 0.6848 (Tong, 1980), RMSE = 0.5727 (Xu and Lu, 1987), RMSE = 0.4494 (Box and Jenkins, 1976) and RMSE = 0.4358 (Sugeno and Yasukawa, 1993).

The results obtained for the  $\epsilon$ -LS-gradient algorithm were compared to the state-of-the-art method based on the Support Vector Regression (SVR) machine (Vapnik, 1998). For this machine the following results were obtained: the Box-Jenkins database—RMSE = 0.3448,  $n_{SV} = 88$  for  $\sigma = 34$ ,  $\epsilon = 0.01$ ,  $C = 1290$ ; the ECG database—RMSE = 0.0263,  $n_{SV} = 215$  for  $\sigma = 2.8$ ,  $\epsilon = 0.02$ ,  $C = 327$ . Taking into account the numbers of support vectors ( $n_{SV}$ ), the following numbers of coefficients were obtained: 1761 for the Box-Jenkins database and 1721 for the ECG database. The best generalization ability of the fuzzy model is obtained for the following numbers of coefficients: 62 and 65, respectively. Thus, we see that the knowledge-base obtained by fuzzy modeling with the  $\epsilon$ -LS-gradient learning algorithm is significantly smaller and, additionally, linguistically interpreted. It can also be noted that fuzzy modeling with the  $\epsilon$ -LS-gradient learning algorithm leads to better generalization for real-world high-dimensional data, compared with the SVR machine. The running times for the ECG database were as follows: the SVR machine—519.9 sec., the fuzzy modeling with the  $\epsilon$ -LS-gradient learning algorithm for 5 if-then rules—5.48 sec.<sup>1</sup> Thus, the running time of fuzzy modeling with  $\epsilon$ -insensitive learning was approximately 100 times shorter with respect to the SVR machine.

## 7. Conclusions

This work presents a new approach to fuzzy modeling with learning tolerant of imprecision. The Vapnik  $\epsilon$ -insensitive loss function is used in this method of learning. It is shown that in this case the problem of  $\epsilon$ -insensitive learning of the consequences of rules for both global and local approaches is equivalent to solving a system of linear inequalities. A modified iterative method for Solving a System of Linear Inequalities ( $\epsilon$ LSSLI1) is also introduced. A hybrid learning method for the premises and consequences of if-then rules is proposed. This method consists of the following steps: First, the parameters of the

<sup>1</sup> The support vector regression (SVR) machine from the Matlab Support Vector Machine Toolbox by S. Gunn was used. This toolbox has been through the Internet—<http://www.isis.ecs.soton.ac.uk/resources/svminfo>.

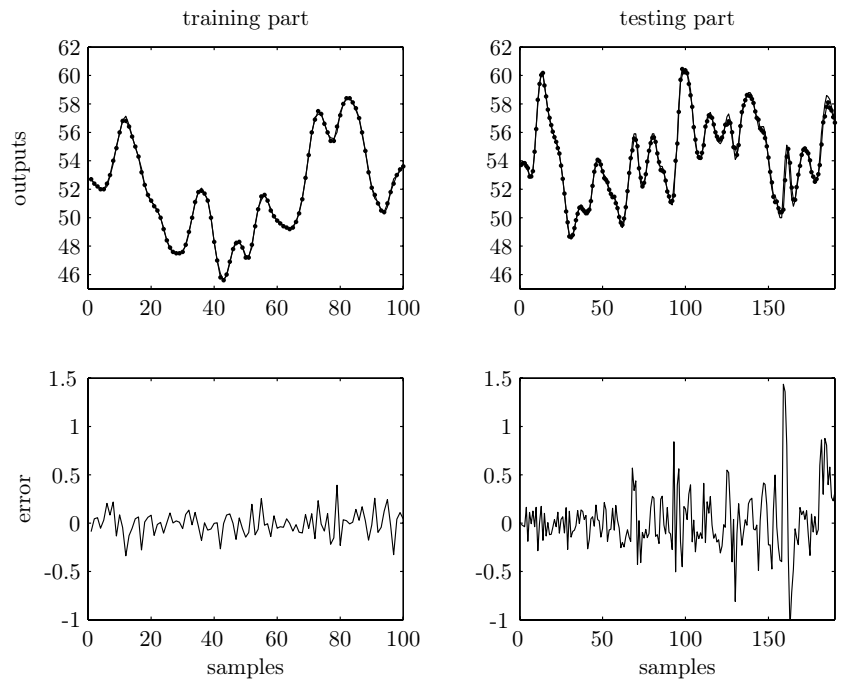


Fig. 1. Box-Jenkins data used in the experiments. The training part on the left and the testing part on the right.

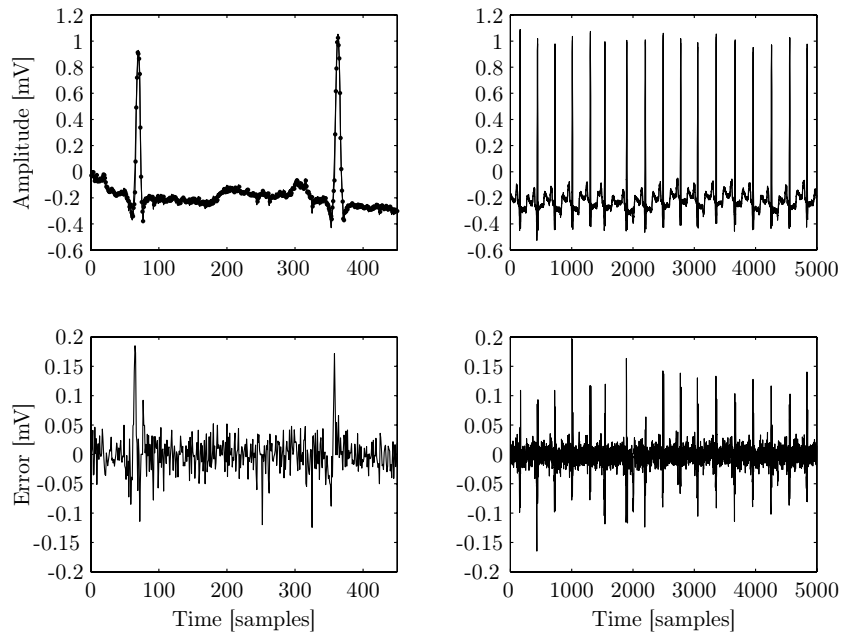


Fig. 2. ECG signal used in the experiments. The training part on the left and the testing part on the right.

premises of rules are obtained using the fuzzy  $c$ -means algorithm. Next, the parameters of the consequents of rules are obtained using  $\varepsilon$ LSSLI1. Then, the above initial parameters of rules are adjusted iteratively. The method of adjusting premises is based on the gradient descent approach. Both squared and  $\varepsilon$ -insensitive functions are used as a loss function.

Numerical examples show the usefulness of the modified iterative method for Solving a System of Linear Inequalities. Given the convergence speed and estimation accuracy, this method outperforms the  $\varepsilon$ LSSLI method. Examples are given of using the proposed hybrid learning of the parameters of the premises and consequence of if-then rules for designing fuzzy models of real-life data. Simulation results show an improvement in the generalization ability of the fuzzy system with respect to the traditional as well as previously introduced  $\varepsilon$ -insensitive learning methods. The experiments also show that the best generalization ability is obtained by means of the hybrid learning method combining the  $\varepsilon$ LSSLI1 method for consequence parameters and the gradient method with a square loss function for premise parameters.

### Acknowledgments

We are grateful to three anonymous referees for their constructive comments that have helped to improve the paper.

### References

- Bezdek J.C. (1982): *Pattern Recognition with Fuzzy Objective Function Algorithms*. — New York: Plenum Press.
- Box G.E.P. and Jenkins G.M. (1976): *Time Series Analysis. Forecasting and Control*. — San Francisco: Holden-Day.
- Castellano G., Fanelli A.M. and Mencor C. (2004): *An empirical risk functional to improve learning in a neuro-fuzzy classifier*. — IEEE Trans. Sys. Man Cybern., Part B: Cybernetics, Vol. 34, No. 1, pp. 725–730.
- Chen J.-Q., Xi Y.-G. and Zhang Z.-J. (1998): *A clustering algorithm for fuzzy model identification*. — Fuzzy Sets Syst., Vol. 98, No. 2, pp. 319–329.
- Chen J.-H. and Chen C.-S. (2002): *Fuzzy kernel perceptron*. — IEEE Trans. Neural Netw., Vol. 13, No. 6, pp. 1364–1373.
- Chiang J.-H. and Hao P.-Y. (2003): *A new kernel-based fuzzy clustering approach: support vector clustering with cell growing*. — IEEE Trans. Fuzzy Syst., Vol. 11, No. 4, pp. 518–527.
- Czogała E. and Łęski J.M. (2000): *Fuzzy and Neuro-Fuzzy Intelligent Systems*. — Heidelberg: Physica-Verlag.
- Czogała E. and Łęski J.M. (2001): *On equivalence of approximate reasoning results using different interpretations of fuzzy if-then rules*. — Fuzzy Sets Syst., Vol. 117, No. 2, pp. 279–296.
- Ho Y.-C. and Kashyap R.L. (1965): *An algorithm for linear inequalities and its applications*. — IEEE Trans. Elec. Comp., Vol. 14, No. 5, pp. 683–688.
- Ho Y.-C. and Kashyap R.L. (1966): *A class of iterative procedures for linear inequalities*. — SIAM J. Contr., Vol. 4, No. 2, pp. 112–115.
- Hong D.H. and Hwang C. (2003): *Support vector fuzzy regression machines*. — Fuzzy Sets Syst., Vol. 138, No. 2, pp. 271–281.
- Jang J.-S.R., Sun C.-T. and Mizutani E. (1997): *Neuro-fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*. — Upper Saddle River: Prentice-Hall.
- Jeng J.-T., Chuang C.-C. and Su S.-F. (2003): *Support vector interval regression networks for interval regression analysis*. — Fuzzy Sets Syst., Vol. 138, No. 2, pp. 283–300.
- Lin C.-F. and Wang S.-D. (2002): *Fuzzy support vector machine*. — IEEE Trans. Neural Netw., Vol. 13, No. 2, pp. 464–471.
- Łęski J.M. and Czogała E. (1999): *A new artificial neural network based fuzzy inference system with moving consequents in if-then rules and its applications*. — Fuzzy Sets Syst., Vol. 108, No. 3, pp. 289–297.
- Łęski J.M. (2001): *An  $\varepsilon$ -insensitive approach to fuzzy clustering*. — Int. J. App. Math. Comp. Sci., Vol. 11, No. 4, pp. 993–1007.
- Łęski J.M. (2001): *Neuro-fuzzy modeling with  $\varepsilon$ -insensitive learning*. — Methods of Artificial Intelligence in Mechanics and Mechanical Engineering, Gliwice, Poland, pp. 133–138.
- Łęski J.M. (2002a):  *$\varepsilon$ -insensitive learning techniques for approximate reasoning systems (Invited Paper)*. — Int. J. Comp. Cognition, Vol. 1, No. 1, pp. 21–77.
- Łęski J.M. (2002b): *Improving generalization ability of neuro-fuzzy system by  $\varepsilon$ -insensitive learning*. — Int. J. Appl. Math. Comp. Sci., Vol. 12, No. 3, pp. 437–447.
- Łęski J.M. (2003a): *Towards a robust fuzzy clustering*. — Fuzzy Sets Syst., Vol. 137, No. 2, pp. 215–233.
- Łęski J.M. (2003b): *Neuro-fuzzy system with learning tolerant to imprecision*. — Fuzzy Sets Syst., Vol. 138, No. 2, pp. 427–439.
- Łęski J.M. (2004a):  *$\varepsilon$ -insensitive fuzzy  $c$ -regression models: Introduction to  $\varepsilon$ -insensitive fuzzy modeling*. — IEEE Trans. Syst. Man Cybern., Part B: Cybernetics, Vol. 34, No. 1, pp. 4–15.
- Łęski J.M. (2004b): *An  $\varepsilon$ -margin nonlinear classifier based on if-then rules*. — IEEE Trans. Syst. Man Cybern., Part B: Cybernetics, Vol. 34, No. 1, pp. 68–76.
- Mendel J.M. (1983): *Optimal Seismic Deconvolution. An Estimation-Based Approach*. — New York: Academic Press.
- Pedrycz W. (1984): *An identification algorithm in fuzzy relational systems*. — Fuzzy Sets Syst., Vol. 13, No. 1, pp. 153–167.



- Rutkowska D. (2001): *Neuro-Fuzzy Architectures and Hybrid Learning*. — Heidelberg: Physica-Verlag.
- Rutkowski L. and Cpalka K. (2003): *Flexible neuro-fuzzy systems*. — IEEE Trans. Neural Netw., Vol. 14, No. 3, pp. 554–574.
- Setnes M. (2000): *Supervised fuzzy clustering for rule extraction*. — IEEE Trans. Fuzzy Syst., Vol. 8, No. 4, pp. 416–424.
- Sugeno M. and Yasukawa T. (1993): *A fuzzy-logic-based approach to qualitative modeling*. — IEEE Trans. Fuzzy Syst., Vol. 1, No. 1, pp. 7–31.
- Tong R.M. (1980): *The evaluation of fuzzy models derived from experimental data*. — Fuzzy Sets Syst., Vol. 4, No. 1, pp. 1–12.
- Vapnik V. (1995): *The Nature of Statistical Learning Theory*. — New York: Springer.
- Vapnik V. (1998): *Statistical Learning Theory*. — New York: Wiley.
- Vapnik V. (1999): *An overview of statistical learning theory*. — IEEE Trans. Neural Netw., Vol. 10, No. 5, pp. 988–999.
- Xie X.L. and Beni G. (1991): *A validity measure for fuzzy clustering*. — IEEE Trans. Pattern Anal. Mach. Intell., Vol. 13, No. 8, pp. 841–847.
- Xu C.W. and Lu Y.Z. (1987): *Fuzzy model identification and self-learning for dynamic systems*. — IEEE Trans. Syst., Man Cybern., Vol. 17, No. 3, pp. 190–197.
- Zadeh L. A. (1973): *Outline of a new approach to the analysis of complex systems and decision processes*. — IEEE Trans. Syst. Man Cybern., Vol. 3, No. 1, pp. 28–44.

Received: 6 October 2004

Revised: 25 January 2005

