Int. J. Appl. Math. Comput. Sci., 2002, Vol.12, No.3, 437–447

IMPROVING THE GENERALIZATION ABILITY OF NEURO-FUZZY SYSTEMS BY ε-INSENSITIVE LEARNING

JACEK ŁĘSKI*

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

A new learning method tolerant of imprecision is introduced and used in neuro-fuzzy modelling. The proposed method makes it possible to dispose of an intrinsic inconsistency of neuro-fuzzy modelling, where zero-tolerance learning is used to obtain a fuzzy model tolerant of imprecision. This new method can be called ε-insensitive learning, where, in order to fit the fuzzy model to real data, the ε-insensitive loss function is used. ε-insensitive learning leads to a model with minimal Vapnik-Chervonenkis dimension, which results in an improved generalization ability of this system. Another advantage of the proposed method is its robustness against outliers. This paper introduces two approaches to solving ε-insensitive learning problem. The first approach leads to a quadratic programming problem with bound constraints and one linear equality constraint. The second approach leads to a problem of solving a system of linear inequalities. Two computationally efficient numerical methods for ε-insensitive learning are proposed. Finally, examples are given to demonstrate the validity of the introduced methods.

Keywords: fuzzy systems, neural networks, tolerant learning, generalization control, robust methods

1. Introduction

Fuzzy modelling allows finding nonlinear models of reality where knowledge is obtained as a set of if-then rules with linguistically interpreted propositions. Fuzzy modelling is based on the premise that human thinking is tolerant of imprecision and the real world is too complicated to be described precisely (Zadeh, 1964; 1973). Presently, fuzzy modelling plays an important role in many engineering fields, such as pattern recognition, control, identification, data mining, and so on (Czogała and Łęski, 2001; Jang *et al.*, 1997; Rutkowska, 2001; Rutkowska and Hayashi, 1999; Rutkowska and Nowicki, 2000; Wang, 1998).

Methods of fuzzy if-then rules extraction can be divided into (a) those obtained from a human expert, (b) those obtained automatically from observed data, usually by means of artificial neural networks incorporated into fuzzy systems. Methods from the first group have great disadvantages: few experts can and/or want to share their knowledge. In methods from the second group, knowledge is acquired automatically by learning algorithms of neural networks. Such a connection of neural networks and fuzzy models is usually called neuro-fuzzy systems. Neuro-fuzzy modelling has an intrinsic inconsistency. It may perform thinking tolerant of imprecision, but neural network learning methods are zero-tolerant of imprecision, that is, they usually 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 neuro-fuzzy modelling presented in this paper is based on the premise that human learning, as well as thinking, is tolerant of imprecision. Hence, a zero loss is assumed for an error less than some pre-set value, denoted by ε. If the error is greater than ε, then the loss increases linearly. The learning method based on this loss function may be called ε-insensitive learning.

In real applications, data from a training set are corrupted by noise and outliers. It follows that the fuzzy system design methods need to be robust. According to Huber (1981), a robust method should have the following properties: (i) it should have a reasonably good accuracy at the assumed model, (ii) small deviations from the model assumptions should impair the performance only by a small amount, (iii) larger deviations from the model assumptions should not cause a catastrophe. In the literature there are many robust loss functions (Huber, 1981). In this work, the ε-insensitive loss function is used, which is a generalization of the absolute error loss function ($\varepsilon = 0$).

It is well-known in approximation theory (Tikhonov regularization) and machine learning (statistical learning theory) 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 a production of a reasonable decision for data previously unseen in the process of training (Haykin, 1999; Vapnik, 1998). Vapnik-Chervonenkis (VC) theory (or statistical learning theory) has recently emerged as a general theory for estimation of dependencies from a finite set of data (Vapnik, 1999). The most important issue in the VC-theory is the Structural Risk Minimization (SRM) induction principle. The SRM principle suggest a trade-off between the quality of an approximation and the complexity of the approximation function (Vapnik, 1998). A measure of the approximation function complexity (or capacity) is called the VC-dimension. One of the simplest methods of controlling the VC-dimension is to change the insensitivity parameter ε in the loss function. Increasing ε results in decreasing VC-dimension.

Although the idea of learning tolerant of imprecision can be incorporated into all fuzzy system design methods, in this work a method based on fuzzy partition of input space will be presented, due to its simplicity. First, Pedrycz (1984) used the fuzzy c-means clustering (introduced by Bezdek (1982)) to find antecedent variable membership functions and then to identify a relational fuzzy model. Many authors (c.f. Chen *et al.*, 1998; Czogała and Łęski, 2001; Setnes, 2000; Wang, 1998) have recently used fuzzy c-means to find clusters in the input space preserving the similarity of input data, where each cluster corresponds to a fuzzy if-then rule in the Takagi-Sugeno-Kang form (Sugeno and Kang, 1988; Takagi and Sugeno, 1985):

$$R^{(i)}:\ \text{IF}\ \ \mathbf{x}\ \text{is}\ \mathbf{A}^{(i)},\ \ \text{THEN}\ \ y=\mathbf{w}^{(i)T}\mathbf{x}', \qquad (1)$$

$i=1,2,\ldots,c$, where $\mathbf{x}\in\mathbb{R}^t$ is the input variable, $y\in\mathbb{R}$ is the output variable, $\mathbf{x}'\triangleq[\mathbf{x}^T\ 1]^T$ is the augmented input vector, $\mathbf{w}^{(i)}=[\widetilde{\mathbf{w}}^{(i)T}w_0^{(i)}]^T\in\mathbb{R}^{t+1}$ is the vector of consequent parameters of the i-th rule, and $w_0^{(i)}$ denotes the bias of the i-th model. $\mathbf{A}^{(i)}$ is the antecedent fuzzy set of the i-th rule, with membership function $\mathbf{A}^{(i)}(\mathbf{x}):\mathbb{R}^t\to[0,1]$. In the case of Gaussian membership functions and the algebraic product as the t-norm, the fuzzy antecedent is defined as

$$\mathbf{A}^{(i)}(\mathbf{x})\triangleq\prod_{j=1}^{t}\exp\left[-\frac{\left(x_j-c_j^{(i)}\right)^2}{2\left(s_j^{(i)}\right)^2}\right]$$
$$=\exp\left[-\frac{1}{2}\sum_{j=1}^{t}\left(\frac{x_j-c_j^{(i)}}{s_j^{(i)}}\right)^2\right], \qquad (2)$$

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

$$c_j^{(i)}=\frac{\sum_{n=1}^{N}u_{in}\,x_{nj}}{\sum_{n=1}^{N}u_{in}} \qquad (3)$$

and

$$s_j^{(i)}=\frac{\sum_{n=1}^{N}u_{in}\left(x_{nj}-c_j^{(i)}\right)^2}{\sum_{n=1}^{N}u_{in}}, \qquad (4)$$

where u_{in} denotes an element of the partition matrix obtained from the fuzzy c-means clustering of the input space.

For the input $\mathbf{x}$, the overall output of the fuzzy model is completed with a weighted averaging aggregation of outputs of individual rules (1) as (Jang *et al.*, 1997; Wang, 1998)

$$y=f(\mathbf{x},\mathbf{W})=\frac{\sum_{i=1}^{c}\mathbf{A}^{(i)}(\mathbf{x})\mathbf{w}^{(i)T}\mathbf{x}'}{\sum_{i=1}^{c}\mathbf{A}^{(i)}(\mathbf{x})}$$
$$=\sum_{i=1}^{c}\overline{\mathbf{A}^{(i)}}(\mathbf{x})\mathbf{w}^{(i)T}\mathbf{x}', \qquad (5)$$

where $\overline{\mathbf{A}^{(i)}}(\mathbf{x})$ is the normalized firing strength of the i-th rule for input $\mathbf{x}$ and $\mathbf{W}=[\mathbf{w}^{(1)T}\mathbf{w}^{(2)T}\cdots\mathbf{w}^{(c)T}]^T$ denotes the consequent parameter vector. It must also be noted that (2) and (5) describe a radial-basis-like neural network. If we define

$$\mathbf{d}(\mathbf{x}')=\left[\overline{\mathbf{A}^{(1)}(\mathbf{x})}\mathbf{x}'^T\vdots\overline{\mathbf{A}^{(2)}(\mathbf{x})}\mathbf{x}'^T\vdots\cdots\vdots\overline{\mathbf{A}^{(c)}(\mathbf{x})}\mathbf{x}'^T\right]^T,$$

then the overall output of the fuzzy system can be written as $y=\mathbf{d}^T(\mathbf{x}')\mathbf{W}$.

Usually, for the consequent parameters $\mathbf{w}^{(i)}$, the Least Squares (LS) estimation is applied (Jang *et al.*, 1997; Setnes, 2000; Sugeno and Kang, 1988). There are two approaches: (a) to solve c independent weighted LS problems, one for each if-then rule, (b) to solve one global LS problem. The first approach leads to a more reliable local performance, while the second one leads to a better global performance. Combining both the approaches is suggested in (Yen *et al.*, 1998). In the present work the second approach (global learning) will be used to introduce the idea of learning tolerant of imprecision.

Suppose that we have the training set $\mathrm{Tr}^{(N)} = \{(\mathbf{x}1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, where N is the data cardinality, and each independent input datum $\mathbf{x}_i \in \mathbb{R}^t$ has the corresponding dependent output datum $y_i \in \mathbb{R}$. For fixed antecedents obtained via the clustering of the input space, the LS solution to the consequent parameter estimation, minimizing the following criterion function (Yen *et al.*, 1998):

$$I(\mathbf{W}) = \sum_{n=1}^{N} \left[y_n - f(\mathbf{x}_n, \mathbf{W})\right]^2 = (\mathbf{y} - \mathbf{X}\mathbf{W})^T(\mathbf{y} - \mathbf{X}\mathbf{W}), \quad (6)$$

can be written in the matrix form as

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \quad (7)$$

where $\mathbf{X} \triangleq [\mathbf{d}(\mathbf{x}'_1)\ \mathbf{d}(\mathbf{x}'_2)\ \cdots\ \mathbf{d}(\mathbf{x}'_N)]^T \in \mathbb{R}^{N\times c(t+1)}$, $\mathbf{y} = [y_1\ y_2\ \ldots\ y_N]^T$.

The aim of this work is twofold: first, to introduce a new learning method tolerant of imprecision in the case of fuzzy system design methods, (2)–(7); next, to investigate the generalization ability of the fuzzy system obtained by means of this learning method for real-world high-dimensional data.

This paper is organized as follows: Section 2 presents an introduction to the ε-insensitive learning method and shows that this approach leads to a quadratic programming problem. Section 3 presents a new numerical method, called incremental learning, to solve the problem of ε-insensitive learning. ε-insensitive learning by solving a system of inequalities, without the need to solve the quadratic programming problem, is introduced in Section 4. Section 5 presents simulation results and a discussion for the fuzzy modelling of real-world high-dimensional data. Finally, conclusions are drawn in Section 6.

2. ε-Insensitivity in Neuro-Fuzzy System Learning

The problem of learning tolerant of imprecision can be presented as determining the consequent parameters $\mathbf{W}$, where the ε-insensitive loss function is used in order to fit the fuzzy model to real data from the training set. For a scalar argument g, the ε-insensitive loss function has the form (Vapnik, 1998)

$$\rceil g \lceil_\varepsilon \triangleq \begin{cases} 0, & |g| \le \varepsilon, \\ |g| - \varepsilon, & |g| > \varepsilon, \end{cases} \quad (8)$$

and for a vector argument $\mathbf{g} = [g_1\ g_2\ \ldots\ g_N]^T$, it can be defined as

$$\rceil \mathbf{g} \lceil_\varepsilon \triangleq \sum_{n=1}^{N} \rceil g_n \lceil_\varepsilon . \quad (9)$$

Using the augmented input vector $\mathbf{x}'$, we seek a linear regression model in the form

$$y = \mathbf{d}^T(\mathbf{x}')\mathbf{W}, \quad \mathbf{W} \in \mathbb{R}^{c(t+1)}, \quad (10)$$

which minimizes the following criterion:

$$\min_{\mathbf{W}\in\mathbb{R}^{c(t+1)}} I(\mathbf{W}) \triangleq \sum_{n=1}^{N} \rceil y_n - \mathbf{d}^T(\mathbf{x}'_n)\mathbf{W} \lceil_\varepsilon + \frac{\tau}{2}\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}}, \quad (11)$$

where $\widetilde{\mathbf{W}}$ is a reduced vector $\mathbf{W}$, with excluded components corresponding to the biases: $\widetilde{\mathbf{W}} = [\widetilde{\mathbf{w}}^{(1)T}\widetilde{\mathbf{w}}^{(2)T}\cdots\widetilde{\mathbf{w}}^{(c)T}]^T$. The second term in (11) is related to the minimization of the Vapnik-Chervonenkis dimension (complexity) of the regression model (Vapnik, 1998). The parameter $\tau \ge 0$ controls the trade-off between the complexity of the regression model and the amount up to which the errors are tolerated.

Taking into account the fact that $\widehat{y}_n = \mathbf{d}^T(\mathbf{x}'_n)\mathbf{W} = \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} + a$, where $\widetilde{\mathbf{d}}(\mathbf{x}) = [\overline{\mathbf{A}^{(1)}(\mathbf{x})}\mathbf{x}^T \vdots \overline{\mathbf{A}^{(2)}(\mathbf{x})}\mathbf{x}^T \vdots \cdots \vdots \overline{\mathbf{A}^{(c)}(\mathbf{x})}\mathbf{x}^T]^T$ and $a = \overline{\mathbf{A}^{(1)}(\mathbf{x})}w_0^{(1)} + \overline{\mathbf{A}^{(2)}(\mathbf{x})}w_0^{(2)} + \cdots + \overline{\mathbf{A}^{(c)}(\mathbf{x})}w_0^{(c)}$ denotes the overall bias, the criterion (11) can be written in the form

$$\min_{\widetilde{\mathbf{W}}\in\mathbb{R}^{ct}, a\in\mathbb{R}} I(\widetilde{\mathbf{W}}, a) \triangleq \sum_{n=1}^{N} \left| y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} - a \right|_\varepsilon + \frac{\tau}{2}\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}}. \quad (12)$$

In a general case, the inequalities $y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} - a \le \varepsilon$ and $\widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} + a - y_n \le \varepsilon$ are not satisfied for all data $(\mathbf{x}_n, y_n)$. If we introduce slack variables $\xi_n^+, \xi_n^- \ge 0$, then for all data $(\mathbf{x}_n, y_n)$ we can write

$$\begin{cases} y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} - a \le \varepsilon + \xi_n^+, \\ \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} + a - y_n \le \varepsilon + \xi_n^-. \end{cases} \quad (13)$$

Using (13), the criterion (12) can be written in the form

$$I(\widetilde{\mathbf{W}}, a) = \frac{1}{\tau}\sum_{n=1}^{N}\left(\xi_n^+ + \xi_n^-\right) + \frac{1}{2}\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}}, \quad (14)$$

and be minimized subject to constraints (13) and $\xi_n^+ \geq 0$, $\xi_n^- \geq 0$. The Lagrangian function of (14) with the above constraints is

$$\begin{aligned} G = {} & \frac{1}{2}\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}} + \frac{1}{\tau}\sum_{n=1}^{N}(\xi_n^+ + \xi_n^-) \\ & - \sum_{n=1}^{N}\lambda_n^+\left(\varepsilon + \xi_n^+ - y_n + \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} + a\right) \\ & - \sum_{n=1}^{N}\lambda_n^-\left(\varepsilon + \xi_n^- + y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} - a\right) \\ & - \sum_{n=1}^{N}(\mu_n^+\xi_n^+ + \mu_n^-\xi_n^-), \end{aligned} \tag{15}$$

where $\lambda_n^+, \lambda_n^-, \mu_n^+, \mu_n^- \geq 0$ are the Lagrange multipliers. The objective is to minimize this Lagrangian with respect to $\mathbf{W}, a, \xi_n^+, \xi_n^-$. It must also be maximized with respect to the Lagrange multipliers. The following optimality conditions (the saddle point of the Lagrangian) are obtained by differentiating (15) with respect to $\widetilde{\mathbf{W}}, a, \xi_n^+, \xi_n^-$ and setting the results to zero:

$$\begin{cases} \dfrac{\partial G}{\partial \widetilde{\mathbf{W}}} = \widetilde{\mathbf{W}} - \displaystyle\sum_{n=1}^{N}\left(\lambda_n^+ - \lambda_n^-\right)\mathbf{x}_n = 0, \\ \dfrac{\partial G}{\partial a} = \displaystyle\sum_{n=1}^{N}\left(\lambda_n^+ - \lambda_n^-\right) = 0, \\ \dfrac{\partial G}{\partial \xi_n^+} = \dfrac{1}{\tau} - \lambda_n^+ - \mu_n^+ = 0, \\ \dfrac{\partial G}{\partial \xi_n^-} = \dfrac{1}{\tau} - \lambda_n^- - \mu_n^- = 0. \end{cases} \tag{16}$$

The last two conditions (16) and the requirements $\mu_n^+, \mu_n^- \geq 0$ imply that $\lambda_n^+, \lambda_n^- \in [0, 1/\tau]$. From the first condition of (16), we obtain the so-called support vector expansion (Vapnik, 1998):

$$\widetilde{\mathbf{W}} = \sum_{n=1}^{N}\left(\lambda_n^+ - \lambda_n^-\right)\mathbf{x}_n, \tag{17}$$

i.e. $\widetilde{\mathbf{W}}$ can be described as a linear combination of some training data called support vectors. Putting conditions (16) in the Lagrangian (15), we get

$$\begin{aligned} G = {} & -\frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{N}(\lambda_n^+ - \lambda_n^-)(\lambda_j^+ - \lambda_j^-)\widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{d}}(\mathbf{x}_j) \\ & - \varepsilon\sum_{n=1}^{N}(\lambda_n^+ + \lambda_n^-) + \sum_{n=1}^{N}(\lambda_n^+ - \lambda_n^-)y_n. \end{aligned} \tag{18}$$

Maximization of (18) with respect to λ_n^+, λ_n^- subject to constraints

$$\begin{cases} \displaystyle\sum_{n=1}^{N}(\lambda_n^+ - \lambda_n^-) = 0, \\ \lambda_n^+, \lambda_n^- \in [0, 1/\tau] \end{cases} \tag{19}$$

is referred to as the Wolfe dual formulation of (15). It is well known from optimization theory that at the saddle point, for each Lagrange multiplier, the Karush-Kuhn-Tucker (KKT) conditions must be satisfied:

$$\begin{cases} \lambda_n^+(\varepsilon + \xi_n^+ - y_n + \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} + a) = 0, \\ \lambda_n^-(\varepsilon + \xi_n^- + y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} - a) = 0, \\ \left(\dfrac{1}{\tau} - \lambda_n^+\right)\xi_n^+ = 0, \\ \left(\dfrac{1}{\tau} - \lambda_n^-\right)\xi_n^- = 0. \end{cases} \tag{20}$$

From the last two conditions of (20), we see that $\lambda_n^+ \in (0, 1/\tau) \implies \xi_n^+ = 0$ and $\lambda_n^- \in (0, 1/\tau) \implies \xi_n^- = 0$. In this case, from the first two conditions of (20), we have

$$\begin{cases} a = y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} - \varepsilon \ \text{ for } \ \lambda_n^+ \in (0, 1/\tau), \\ a = y_n - \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{W}} + \varepsilon \ \text{ for } \ \lambda_n^- \in (0, 1/\tau). \end{cases} \tag{21}$$

Thus we can determine the parameter a from (21) by taking any $\mathbf{x}_n$ for which there are Lagrange multipliers in the open interval $(0, 1/\tau)$. From a numerical point of view, it is better to take the mean value of a obtained for all data for which the conditions from (21) are satisfied. Taking into account that $a = \overline{\mathbf{A}^{(1)}(\mathbf{x})}w_0^{(1)} + \overline{\mathbf{A}^{(2)}(\mathbf{x})}w_0^{(2)} + \cdots + \overline{\mathbf{A}^{(c)}(\mathbf{x})}w_0^{(c)}$ and $\sum_{i=1}^{c}\overline{\mathbf{A}^{(i)}(\mathbf{x})} = 1$, we see that $w_0^{(i)} = a,\ i = 1, 2, \ldots, c$.

Computation of the parameters $\widetilde{\mathbf{W}}$ and a leads to the quadratic programming (QP) problem (18) with bound constraints and one linear equality constraint (19). For a large training set, standard optimization techniques quickly become intractable in their memory and time requirements. Standard implementation of QP solvers requires the explicit storage of $N \times N$ matrix. Osuna *et al.* (1997) and Joachims (1999) show that large QP problems can be decomposed into a series of smaller QP subproblems over part of data. Platt (1999) proposes the Sequential Minimal Optimization algorithm. This method chooses two Lagrange multipliers and finds their optimal values analytically. A disadvantage of these techniques is that they may give an approximate solution and may require many passes through the training set. In (Cauwenberghs and Poggio, 2001) an alternative approach that determines the exact solution for p training data pairs in terms of that for $p-1$ data pairs to solve classification problems is presented. In the next section this idea is used to solve the problem of fuzzy modelling.

3. Incremental Learning

Putting the first and the last two conditions of (16) in the Lagrangian (15), we get

$$H = -G = \frac{1}{2}\sum_{n=1}^{N} x \sum_{j=1}^{N}(\lambda_n^+ - \lambda_n^-)(\lambda_j^+ - \lambda_j^-)\widetilde{\mathbf{d}}^T(\mathbf{x}_n)\widetilde{\mathbf{d}}(\mathbf{x}_j)$$
$$+ \varepsilon \sum_{n=1}^{N}(\lambda_n^+ + \lambda_n^-) - \sum_{n=1}^{N}(\lambda_n^+ - \lambda_n^-)y_n$$
$$- a\sum_{n=1}^{N}(\lambda_n^+ - \lambda_n^-). \tag{22}$$

Defining $\lambda_n^{\pm} \triangleq \lambda_n^+ - \lambda_n^- \in [-1/\tau, +1/\tau]$ the minimization of (22) can be written in the form

$$\min_{\{-1/\tau \le \lambda_n^{\pm} \le +1/\tau\}, a \in \mathbb{R}} H$$
$$= \frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{N}\lambda_n^{\pm}\lambda_j^{\pm}K_{nj} + \varepsilon\sum_{n=1}^{N}\left|\lambda_n^{\pm}\right|$$
$$- \sum_{n=1}^{N}\lambda_n^{\pm}y_n - a\sum_{n=1}^{N}\lambda_n^{\pm}, \tag{23}$$

where $K_{nj} \triangleq \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\,\widetilde{\mathbf{d}}(\mathbf{x}_j)$ is the dot-product kernel.

Differentiating (23) with respect to $\lambda_n^{\pm}$ and a yields

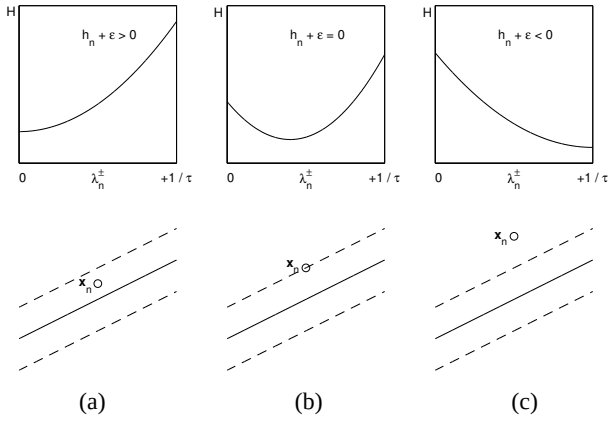
$$\begin{cases} \dfrac{\partial H}{\partial \lambda_n^{\pm}} = \displaystyle\sum_{j=1}^{N} K_{nj}\lambda_j^{\pm} + a + \epsilon\,\mathrm{sgn}(\lambda_n^{\pm}) - y_n, \\ \dfrac{\partial H}{\partial a} = \displaystyle\sum_{j=1}^{N}\lambda_j^{\pm}. \end{cases} \tag{24}$$

Using (17), we see that the first equation of (24) is the KKT condition. By defining $h_n \triangleq \displaystyle\sum_{j=1}^{N} K_{nj}\lambda_j^{\pm} + a - y_n = \widetilde{\mathbf{d}}^T(\mathbf{x}_n)\,\widetilde{\mathbf{W}} + a - y_n$, the following conditions are satisfied:

- if $h_n > 0$, then $\mathbf{x}_n$ is below the regression line,
- if $h_n = 0$, then $\mathbf{x}_n$ is on the regression line,
- if $h_n < 0$, then $\mathbf{x}_n$ is above the regression line,

and

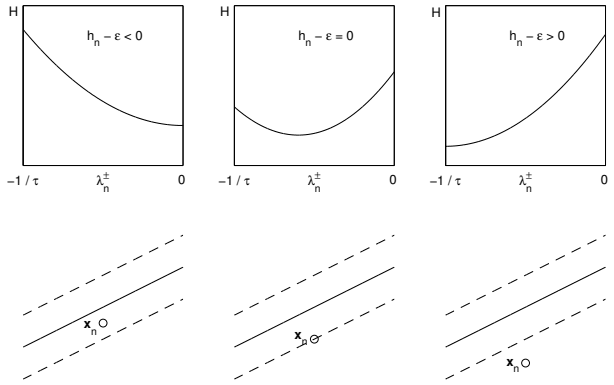
- if $h_n + \epsilon < 0$, then $\mathbf{x}_n$ is above the insensitivity region, and $\lambda_n^{\pm} = +1/\tau$, see Fig. 1(c),
- if $h_n + \epsilon = 0$, then $\mathbf{x}_n$ is the support vector of the regression, and $\lambda_n^{\pm} \in (0, +1/\tau)$, see Fig. 1(b),
- if $h_n + \epsilon > 0$, then $\mathbf{x}_n$ is on the insensitivity region, and $\lambda_n^{\pm} \to 0^+$, see Fig. 1(a),



(a) (b) (c)

Fig. 1. The Karush-Kuhn-Tucker conditions for points above the regression line.

- if $h_n - \epsilon < 0$, then $\mathbf{x}_n$ is on the insensitivity region, and $\lambda_n^{\pm} \to 0^-$, see Fig. 2(a),
- if $h_n - \epsilon = 0$, then $\mathbf{x}_n$ is the support vector of the regression, and $\lambda_n^{\pm} \in (-1/\tau, 0)$, see Fig. 2(b),
- if $h_n - \epsilon > 0$, then $\mathbf{x}_n$ is below the insensitivity region, and $\lambda_n^{\pm} = -1/\tau$, see Fig. 2(c).



(a) (b) (c)

Fig. 2. The Karush-Kuhn-Tucker conditions for points below the regression line.

The parameters $\{\lambda_n^{\pm}, a\}$ explicitly define the partition of the training data $\mathrm{Tr}^{(N)}$ into the following groups: support vector set $\mathrm{Sv}^{(N)}$ (Fig. 1(b) and Fig. 2(b)), error vectors set $\mathrm{Er}^{(N)}$ (Fig. 1(c) and Fig. 2(c)), and remaining vectors set $\mathrm{Re}^{(N)}$ (Fig. 1(a) and Fig. 2(a)), where $\mathrm{Sv}^{(N)} \cup \mathrm{Er}^{(N)} \cup \mathrm{Re}^{(N)} = \mathrm{Tr}^{(N)}$. In incremental learning the solution in the iteration p is obtained from the solution in the iteration $p-1$. In the iteration p the data pair $(\mathbf{x}_c, y_c)$ is added to the training set $\mathrm{Tr}^{(p)} = \mathrm{Tr}^{(p-1)} \cup \{(\mathbf{x}_c, y_c)\}$. First, we assume that initially $\lambda_c^{\pm}$ is equal to zero, and it is changed by a small value $\partial\lambda_c^{\pm}$. The regression parameters $\widetilde{\mathbf{W}}$ and a change their values in

each incremental step to keep all elements from the training set in equilibrium, i.e. to keep the KKT conditions fulfilled. These conditions can be differentially expressed as

$$\begin{cases} \partial h_n = K_{nc}\partial\lambda_c^{\pm} + \sum\limits_{j\in \mathrm{Sv}^{(p-1)}} K_{nj}\partial\lambda_j^{\pm} + \partial a = 0, \\ \partial\lambda_c^{\pm} + \sum\limits_{j\in \mathrm{Sv}^{(p-1)}} \partial\lambda_j^{\pm} = 0. \end{cases} \tag{25}$$

For vectors from the support set $\mathrm{Sv}^{(p-1)} = \{(\mathbf{x}_{s_1}, y_{s_1}), (\mathbf{x}_{s_2}, y_{s_2}), \ldots, (\mathbf{x}_{s_\ell}, y_{s_\ell})\}$, the following conditions are fulfilled: $\partial h_{s_k} = 0,\ 1 \le k \le \ell$. Writing (25) for $\mathrm{Sv}^{(p-1)}$ in the matrix form yields

$$\Gamma \begin{bmatrix} \partial a \\ \partial\lambda_{s_1}^{\pm} \\ \vdots \\ \partial\lambda_{s_\ell}^{\pm} \end{bmatrix} = - \begin{bmatrix} 1 \\ K_{s_1 c} \\ \vdots \\ K_{s_\ell c} \end{bmatrix} \partial\lambda_c^{\pm}, \tag{26}$$

where Γ is the symmetric non-positive definite Jacobian

$$\Gamma = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K_{s_1 s_1} & \cdots & K_{s_1 s_\ell} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{s_\ell s_1} & \cdots & K_{s_\ell s_\ell} \end{bmatrix}. \tag{27}$$

In the equilibrium state, we have

$$\begin{cases} \partial a = \psi\partial\lambda_c^{\pm}, \\ \partial\lambda_n^{\pm} = \rho_n\partial\lambda_c^{\pm}, \end{cases} \tag{28}$$

with sensitivities given by

$$\begin{bmatrix} \psi \\ \rho_{s_1} \\ \vdots \\ \rho_{s_\ell} \end{bmatrix} = -\Upsilon \begin{bmatrix} 1 \\ K_{s_1 c} \\ \vdots \\ K_{s_\ell c} \end{bmatrix}, \tag{29}$$

where $\Upsilon = \Gamma^{-1}$, and for all $\mathbf{x}_j$ outside $\mathrm{Sv}^{(p-1)}$ we have $\rho_j = 0$. Substituting (28) in ∂h_n yields

$$\partial h_n = \kappa_n \partial\lambda_c^{\pm}, \tag{30}$$

where $\kappa_n = 0$ for all $\mathbf{x}_n \in \mathrm{Sv}^{(p-1)}$, and

$$\kappa_n = K_{nc} + \sum_{j\in \mathrm{Sv}^{(p-1)}} K_{nj}\rho_j + \psi, \quad \text{for } \mathbf{x}_n \notin \mathrm{Sv}^{(p-1)}. \tag{31}$$

If $\partial\lambda_c^{\pm}$ is sufficiently large, then elements of $Tr^{(p-1)}$ move across the sets $Sv^{(p-1)}$, $Er^{(p-1)}$, $Re^{(p-1)}$. On the basis of (28) and (30), it is possible to determine the largest admissible value of $\partial\lambda_c^{\pm}$ to the first membership change according to

- $h_c + \varepsilon \le 0$, with equality when $(\mathbf{x}_c, y_c)$ joins to $\mathrm{Sv}^{(p-1)}$,
- $h_c - \varepsilon \ge 0$, with equality when $(\mathbf{x}_c, y_c)$ joins to $\mathrm{Sv}^{(p-1)}$,
- $\lambda_c^{\pm} \le +1/\tau$, with equality when $(\mathbf{x}_c, y_c)$ joins to $\mathrm{Er}^{(p-1)}$,
- $\lambda_c^{\pm} \ge -1/\tau$, with equality when $(\mathbf{x}_c, y_c)$ joins to $\mathrm{Er}^{(p-1)}$,
- $0 \le \lambda_n^{\pm} \le +1/\tau,\ \forall n \in \mathrm{Sv}^{(p-1)}$, with equality to 0, when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Sv}^{(p-1)}$ to $\mathrm{Re}^{(p-1)}$, and equality to $+1/\tau$, when $(\mathbf{x}_n, y_n)$ transfers from $\mathrm{Sv}^{(p-1)}$ to $\mathrm{Er}^{(p-1)}$,
- $-1/\tau \le \lambda_n^{\pm} \le 0,\ \forall n \in \mathrm{Sv}^{(p-1)}$, with equality to $-1/\tau$, when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Sv}^{(p-1)}$ to $\mathrm{Er}^{(p-1)}$, and equality to 0, when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Sv}^{(p-1)}$ to $\mathrm{Re}^{(p-1)}$,
- $h_n + \varepsilon \le 0,\ \forall n \in \mathrm{Er}^{(p-1)}$, with equality when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Er}^{(p-1)}$ to $\mathrm{Sv}^{(p-1)}$,
- $h_n - \varepsilon \ge 0,\ \forall n \in \mathrm{Re}^{(p-1)}$, with equality when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Re}^{(p-1)}$ to $\mathrm{Sv}^{(p-1)}$,
- $h_n + \varepsilon \ge 0,\ \forall n \in \mathrm{Re}^{(p-1)}$, with equality when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Re}^{(p-1)}$ to $\mathrm{Sv}^{(p-1)}$,
- $h_n - \varepsilon \le 0,\ \forall n \in \mathrm{Re}^{(p-1)}$, with equality when $(\mathbf{x}_n, y_k)$ transfers from $\mathrm{Re}^{(p-1)}$ to $\mathrm{Sv}^{(p-1)}$.

If the support vector set is extended by inclusion of an element $s_{\ell+1}$, then the matrix Υ should be extended, too. The matrix Γ is extended by adding one row and one column:

$$\Gamma^{(p)} = \begin{bmatrix} & & & 1 \\ & \Gamma^{(p-1)} & & K_{s_1 s_{\ell+1}} \\ & & & \vdots \\ 1 & K_{s_{\ell+1} s_1} & \cdots & K_{s_{\ell+1} s_{\ell+1}} \end{bmatrix}. \tag{32}$$

Using for (32) the extension principle (from matrix theory (Gantmacher, 1959)) and (29), (31) yields

$$\Upsilon^{(p)} = \begin{bmatrix} & & & 0 \\ & \Upsilon^{(p-1)} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\kappa_{s_{\ell+1}}} \begin{bmatrix} \overrightarrow{\rho} \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{\rho} & 1 \end{bmatrix}, \tag{33}$$

where $\overrightarrow{\rho} = [\psi\rho_1 \cdots \rho_{s_\ell}]$. The extension principle is formulated as follows:

$$\begin{bmatrix} A & u \\ v^T & \beta \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + \frac{A^{-1}uv^TA^{-1}}{\beta'} & -\frac{A^{-1}u}{\beta'} \\ -\frac{v^TA^{-1}}{\beta'} & \frac{1}{\beta'} \end{bmatrix},$$

where $\beta' = \beta - v^TA^{-1}u$. In our case, $A = \Gamma^{(p-1)}$, $A^{-1} = \Upsilon^{(p-1)}$, $v^T = [1K_{s_{\ell+1}s_1} \cdots K_{s_{\ell+1}s_\ell}]$, $u = [1K_{s_1s_{\ell+1}} \cdots K_{s_\ell s_{\ell+1}}]^T$ and $\beta = K_{s_{\ell+1}s_{\ell+1}}$.

The above process is reversible and when the support vector $s_{\ell+1}$ is excluded, we can use (33) to obtain $\Upsilon^{(p-1)}$. In the case of excluding the vector s_j, $j \neq \ell+1$, we can change the position of this vector to $\ell+1$, and use (33):

$$\underset{\substack{m,n\neq j;\, m,n\in \mathrm{Sv}_i^{(p-1)} \\ m,n\neq 0}}{\forall} \Upsilon_{mn}^{(p-1)} = \Upsilon_{mn}^{(p)} - \left[\Upsilon_{mj}^{(p)}\Upsilon_{jn}^{(p)}\right] \Big/ \Upsilon_{jj}^{(p)}. \tag{34}$$

The incremental learning algorithm can be summarized in the following steps:

1. Initialize $h_n = -y_n$, $\lambda_n^{\pm} = 0$, $\forall n$, $\mathrm{Sv}^{(0)} = \mathrm{Er}^{(0)} = \mathrm{Re}^{(0)} = \emptyset$, $p = 1$.
2. Select the element with index c as the farthest one from the regression line.
3. If $h_c + \varepsilon > 0$ or $h_c - \varepsilon < 0$, then $\mathrm{Re}^{(p)} = \mathrm{Re}^{(p-1)} \cup \{(\mathbf{x}_c, y_c)\}$.
4. If $h_c + \varepsilon \leq 0$, then increase $\lambda_c^{\pm}$ so that one of the conditions is fulfilled:
   - $h_c + \varepsilon = 0$, $\mathrm{Sv}^{(p)} = \mathrm{Sv}^{(p-1)} \cup \{(\mathbf{x}_c, y_c)\}$, update $\Upsilon^{(p)}$,
   - $\lambda_c^{\pm} = +1/\tau$, $\mathrm{Er}^{(p)} = \mathrm{Er}^{(p-1)} \cup \{(\mathbf{x}_c, y_c)\}$,
   - One element from $\mathrm{Tr}^{(p-1)}$ is transferred between $\mathrm{Sv}^{(p-1)}$, $\mathrm{Er}^{(p-1)}$, $\mathrm{Re}^{(p-1)}$,
5. If $h_c + \varepsilon \geq 0$, then decrease $\lambda_c^{\pm}$ so that one of the conditions is fulfilled:
   - $h_c + \varepsilon = 0$, $\mathrm{Sv}^{(p)} = \mathrm{Sv}^{(p-1)} \cup \{(\mathbf{x}_c, y_c)\}$, update $\Upsilon^{(p)}$,
   - $\lambda_c^{\pm} = -1/\tau$, $\mathrm{Er}^{(p)} = \mathrm{Er}^{(p-1)} \cup \{(\mathbf{x}_c, y_c)\}$,
   - One element from $\mathrm{Tr}^{(p-1)}$ is transferred between $\mathrm{Sv}^{(p-1)}$, $\mathrm{Er}^{(p-1)}$, $\mathrm{Re}^{(p-1)}$.
6. Update $\lambda_c^{\pm} \longleftarrow \lambda_c^{\pm} + \partial\lambda_c^{\pm}$; $\lambda_n^{\pm} \longleftarrow \lambda_n^{\pm} + \rho_n\partial\lambda_c^{\pm}$, $(\mathbf{x}_n, y_n) \in \mathrm{Sv}^{(p-1)}$; $a \longleftarrow a + \psi\partial\lambda_c^{\pm}$; $h_n \longleftarrow h_n + \kappa_n\partial\lambda_c^{\pm}$, $(\mathbf{x}_n, y_n) \notin \mathrm{Sv}^{(p-1)}$,
7. If there are no processed elements in $\mathrm{Tr}^{(N)}$, then $p \longleftarrow p+1$, go to Step 2.

Remark 1.

- In essence, it is not important which data pair is selected as $(\mathbf{x}_c, y_c)$. But to increase the convergence of the incremental learning, the data pair farthest from the regression line is selected.
- The incremental learning has the computational burden approximately quadratic with the cardinality of the training set. It also is approximately linear with respect to the data dimensionality t.
- Some data pairs are selected several times and, from a computational point of view, it is profitable to cache the kernel evaluations.

4. ε-Insensitive Learning by Solving a System of Linear Inequalities

Let $\widetilde{\mathbf{I}}$ be the identity matrix with the elements on the main diagonal corresponding to the bias elements equal to zero, $\widetilde{\mathbf{I}} = \mathrm{diag}\,([\mathbf{u}^T 0 \vdots \mathbf{u}^T 0 \vdots \cdots \vdots \mathbf{u}^T 0]^T$. Criterion (11) can be rewritten in the matrix form:

$$\min_{\mathbf{W}\in\mathbb{R}^{c(t+1)}} I(\mathbf{W}) \triangleq \rceil \mathbf{y} - \mathbf{X}\mathbf{W} \lceil_{\varepsilon} + \frac{\tau}{2}\mathbf{W}^T\widetilde{\mathbf{I}}\mathbf{W}. \tag{35}$$

To make the minimization problem (35) mathematically tractable, we see that the minimization of the first term can be equivalently written as the requirements $\mathbf{X}\mathbf{W} + \varepsilon\mathbf{u} > y$ and $\mathbf{X}\mathbf{W} - \varepsilon\mathbf{u} < y$, where $\mathbf{u}$ is the vector of ones of the appropriate dimension. Defining the extended versions of $\mathbf{X}$ and $\mathbf{y}$: $\mathbf{X}_e \triangleq [\mathbf{d}(\mathbf{x}_1')\,\mathbf{d}(\mathbf{x}_2') \cdots \mathbf{d}(\mathbf{x}_N') \vdots -\mathbf{d}(\mathbf{x}_1') - \mathbf{d}(\mathbf{x}_2') \cdots -\mathbf{d}(\mathbf{x}_N')]^T$ and $\mathbf{y}_e \triangleq [y_1 - \varepsilon\, y_2 - \varepsilon \;\cdots\; y_N - \varepsilon \;\vdots\; -y_1 - \varepsilon\; -y_2 - \varepsilon \;\cdots\; -y_N - \varepsilon]^T$, the above requirements can be written as $\mathbf{X}_e\mathbf{W} - \mathbf{y}_e > \mathbf{0}$. In practically interesting cases, not all inequalities in the above system are fulfilled (except for the case where ε is so large that all data fall in the insensitivity region). In computations, the above inequality system is replaced by the equality system $\mathbf{X}_e\mathbf{W} - \mathbf{y}_e = \mathbf{b}$, where $\mathbf{b}$ is an arbitrary positive vector $\mathbf{b} > \mathbf{0}$.

We define the error vector as $\mathbf{e} = \mathbf{X}_e\mathbf{W} - \mathbf{y}_e - \mathbf{b}$. If the p-th ($2p$-th) component of $\mathbf{e}$, $1 \leq p \leq N$, is positive, $e_p \geq 0$ ($e_{2p} \geq 0$), then the p-th datum falls into the insensitivity region, and by increasing the respective component of $\mathbf{b}$, e_p (e_{2p}) can be set to zero. If the p-th ($2p$-th) component of $\mathbf{e}$ is negative, then the p-th datum falls outside the insensitivity region and it is impossible to decrease b_p (b_{2p}) and to fulfil the condition $b_p > 0$ ($b_{2p} > 0$). In other words, we obtain a non-zero error only for a datum outside the insensitivity region. Our

minimization problem (35) can be approximated by the following:

$$\min_{\mathbf{W}\in\mathbb{R}^{c(t+1)},\mathbf{b}>\mathbf{0}} I(\mathbf{W},\mathbf{b}) \triangleq \left(\mathbf{X}_e\mathbf{W}-\mathbf{y}_e-\mathbf{b}\right)^T \times \mathbf{D}_e\left(\mathbf{X}_e\mathbf{W}-\mathbf{y}_e-\mathbf{b}\right)+\frac{\tau}{2}\mathbf{W}^T\widetilde{\mathbf{I}}\mathbf{W}, \quad (36)$$

where $\mathbf{D}_e$ denotes a diagonal weighting matrix. For mathematical simplicity, the above criterion is an approximation of (35), where the squared error rather than the absolute error is used. In what follows, the absolute error will be used.

We obtain optimality conditions by differentiating (36) with respect to $\mathbf{W}$ and $\mathbf{b}$, and by setting the results to zero:

$$\begin{cases} \mathbf{W}=\left(\mathbf{X}_e^T\mathbf{D}_e\mathbf{X}_e+\frac{\tau}{2}\widetilde{\mathbf{I}}\right)^{-1}\mathbf{X}_e^T\mathbf{D}_e\left(\mathbf{y}_e+\mathbf{b}\right), \\ \mathbf{e}=\mathbf{X}_e\mathbf{W}-\mathbf{y}_e-\mathbf{b}=\mathbf{0}. \end{cases} \quad (37)$$

From the first equation of (37), we see that the vector $\mathbf{W}$ depends on the vector $\mathbf{b}$. The vector $\mathbf{b}$ can be called a margin vector because its components determine the distance from a datum to the insensitivity region. For fixed $\mathbf{W}$, if a datum is placed in the insensitivity region, the corresponding distance can be increased in order to obtain the zero error. However, if a datum is placed outside the insensitivity region, then the error is negative and we can decrease the error only by decreasing the corresponding component of $\mathbf{b}$. The only way to prevent $\mathbf{b}$ from converging to zero is to start with $\mathbf{b}>\mathbf{0}$ and to refuse to decrease any of its components. Ho and Kashyap proposed an iterative algorithm for alternately determining $\mathbf{W}$ and $\mathbf{b}$, where components of $\mathbf{b}$ cannot decrease (Ho and Kashyap, 1965, Ho and Kashyap, 1966). Now, this algorithm can be extended to our weighted squared error criterion with the term corresponding to the VC dimension. The solution is obtained in an iterative way. Vector $\mathbf{W}$ is determined on the basis of the first equation of (37), i.e. $\mathbf{W}^{[k]}=(\mathbf{X}_e^T\mathbf{D}_e\mathbf{X}_e+\tau/2\widetilde{\mathbf{I}})^{-1}\mathbf{X}_e^T\mathbf{D}_e(\mathbf{y}_e+\mathbf{b}^{[k]})$, where superscript $[k]$ denotes the iteration index. Components of vector $\mathbf{b}$ are modified by components of the error vector $\mathbf{e}$, but only in the case when it results in increasing components of $\mathbf{b}$. Otherwise, the components of $\mathbf{b}$ remain unmodified. So, we write this modification as follows:

$$\mathbf{b}^{[k+1]}=\mathbf{b}^{[k]}+\rho\left(\mathbf{e}^{[k]}+\left|\mathbf{e}^{[k]}\right|\right), \quad (38)$$

where $\rho>0$ is a parameter.

The absolute error criterion, equivalent to (35), is easily obtained by selecting the diagonal weighting matrix $\mathbf{D}_e=\mathrm{diag}(1/|e_1|,\ldots,1/|e_{2N}|)$, where e_i is the i-th component of the error vector. However, the error vector depends on $\mathbf{W}$, so we use vector $\mathbf{W}$ from the previous iteration. This procedure is based on the premise that sequential vectors $\mathbf{W}$ differ imperceptibly near the optimal solution. The procedure of looking for optimal $\mathbf{W}$ and $\mathbf{b}$ may be called iterative learning and can be summarized in the following steps:

1. Fix $\tau\geq 0$, $\rho>0$ and $\mathbf{D}_e^{[1]}=\mathrm{diag}(\mathbf{u}^T)$. Initialize $\mathbf{b}^{[1]}>\mathbf{0}$. Set iteration index $k=1$.
2. Set $\mathbf{W}^{[k]}=(\mathbf{X}_e^T\mathbf{D}_e^{[k]}\mathbf{X}_e+\tau/2\widetilde{\mathbf{I}})^{-1}\mathbf{X}_e^T\mathbf{D}_e^{[k]}(\mathbf{y}_e+\mathbf{b}^{[k]})$.
3. Set $\mathbf{e}^{[k]}=\mathbf{X}_e\mathbf{W}^{[k]}-\mathbf{y}_e-\mathbf{b}^{[k]}$.
4. Set $\mathbf{D}_e^{[k+1]}=\mathrm{diag}\,(1/|e_1^{[k]}|,\ldots,1/|e_{2N}^{[k]}|)$.
5. Set $\mathbf{b}^{[k+1]}=\mathbf{b}^{[k]}+\rho[\mathbf{e}^{[k]}+|\mathbf{e}^{[k]}|]$.
6. If $(\|\mathbf{b}^{[k+1]}-\mathbf{b}^{[k]}\|>\xi)$ and $(k<k_{\max})$, then $k=k+1$, go to Step 2.

Remark 2. ξ is a pre-set parameter. The maximal number of iterations is denoted by $k_{\max}$. Appendix A shows that for $0<\rho<1$ and any diagonal matrix $\mathbf{D}_e$, the above algorithm is convergent. If Step 4 in this algorithm is omitted, then the squared error minimization procedure is obtained. In practice, the divide-by-zero-error in Step 4 does not occur. This follows from the fact that some components of vector $\mathbf{e}$ tend to zero as $[k]$ tends to infinity. But in this case the convergence is slow and Condition 6 stops the algorithm.

5. Numerical Experiments and Discussion

In all experiments $\mathbf{b}^{[1]}=10^{-6}$, $\rho=0.98$ and $k_{\max}=100$ were used in the iterative learning algorithm. The iterations were stopped when the Euclidean norm in a successive pair of $\mathbf{b}$ vectors was less than 10^{-4}. For both the incremental and the iterative learning the fuzzy c-means clustering algorithm was applied with the weighted exponent equal to 2. A random partition matrix was used for initialization, and iterations were stopped when the Frobenius norm in a successive pair of partition matrices was less than 10^{-6}. All experiments were run in the MATLAB® environment.

The purpose of these experiments was to compare the generalization ability of the proposed neuro-fuzzy system with learning tolerant of imprecision and the classical (zero-tolerant) learning. The following benchmark databases were used:

- Data originating from Box and Jenkins' work (1976) 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_2 (percentage content—output signal $y(n)$). The data consisted of 296 pairs of input-output samples with the sampling period of 9 s. To identify the model, the vectors $\mathbf{x}_n \triangleq [y(n-1)\dots y(n-4)\,x(n)\,x(n-1)\dots x(n-6)]^T$ and $y_n \triangleq y(n)$ were used as the input and output, respectively. The learning set consists of the first 100 pairs of data and the testing set consists of the remaining 190 pairs of data.

- Data originating from (Weigend *et al.*, 1990) concerning the prediction of the number of sunspots. The data consisted of 280 measurements of sunspot activity $x(n)$ with a one-year sampling period, from 1700 to 1979 A.D. To identify the model, the vectors $\mathbf{x}_n \triangleq [x(n-1)\,x(n-2)\,x(n-3)\dots x(n-12)]^T$, $y_n \triangleq x(n)$ were used as the input and output, respectively. The learning set consists of the first 100 vectors and the testing set consists of the remaining 168 vectors.

Parameters τ and ε were changed in the range from 0 to 0.5 (step 0.01) and the number of if-then rules was changed from 2 to 6. After the training stage (neuro-fuzzy system design on the training set), 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 25 times for different random initializations of the FCM method, for each combination of the above parameter values. Tables 1 and 2 show the lowest RMSE for each number of if-then rules for Box-Jenkins and Sunspot databases, respectively. Also values of τ and ε parameters for which the lowest RMSE was obtained as well as RMSE for zero-tolerant learning are shown.

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 a better generalization compared with zero-tolerant learning, for both the databases. The best generalization for each number of rules is obtained for non-zero parameters ε and τ. It must also be noted that we observe different dependencies of the generalization ability on the number of if-then rules for zero-tolerant learning and learning tolerant of imprecision. For zero-tolerant learning, increasing the number of rules results in decreasing fast the generalization ability due to the overfitting effect of the training set. For both the learning methods tolerant of imprecision we have a slower decrease in of the generalization ability. The best generalization ability for Box-Jenkins data is obtained using the iterative learning algorithm with $c = 2$, $\varepsilon = 0.02$ and $\tau = 0.02$. However, for the Sunspot database the best generalization is obtained for the incremental learning algorithm with $c = 3$, $\varepsilon = 0.07$ and $\tau = 0.1$.

Table 1. RMSE obtained for the testing part of the Box-Jenkins time series.

c	Incremental learning			Iterative learning			Zero-tolerant learning
	RMSE	ε	τ	RMSE	ε	τ	RMSE
2	0.3533	0.01	0.25	0.3455	0.02	0.02	0.3622
3	0.3731	0.10	0.20	0.3582	0.09	0.02	0.3785
4	0.3722	0.01	0.08	0.3730	0.02	0.32	0.4349
5	0.3886	0.10	0.10	0.3519	0.02	0.16	0.4272
6	0.5298	0.01	0.01	0.4956	0.02	0.48	0.5395

Table 2. RMSE obtained for the testing part of the Sunspot database.

c	Incremental learning			Iterative learning			Zero-tolerant learning
	RMSE	ε	τ	RMSE	ε	τ	RMSE
2	0.08945	0.10	0.10	0.08676	0.12	0.42	0.09641
3	0.08010	0.07	0.10	0.08021	0.08	0.18	0.08569
4	0.08612	0.07	0.07	0.08116	0.08	0.20	0.1112
5	0.1028	0.06	0.07	0.09331	0.01	0.40	0.1164
6	0.08434	0.05	0.08	0.08823	0.05	0.32	0.1404

Figure 3 illustrates the performance of learning tolerant of imprecision (the iterative learning with $c = 2$, $\varepsilon = 0.02$, $\tau = 0.02$) for the Box-Jenkins data. In this figure, the output (solid line) of the original data and the output of the model (dotted line) are shown. Figure 4 illustrates the performance of learning tolerant of imprecision (the incremental learning with $c = 3$, $\varepsilon = 0.07$, $\tau = 0.10$) for the Sunspot database. In this figure, the output (solid line) of the original data and the output of the model (dotted line) are shown.

Finally, it can be noted that the proposed iterative learning algorithm tolerant of imprecision converges after a few iterations, but its computational burden is approximately four times bigger in comparison with zero-tolerant learning. The computational burden of incremental learning is approximately seven times bigger in comparison with zero-tolerant learning.

A simple test for the robustness of the proposed learning method against outliers was also performed. For the Box-Jenkins database, the first output sample y_1, which has the original value equal to 52.7, was set to 100. Hence only one outlier was added to the database. In this case, zero-tolerant learning leads to the RMSE for the testing part of the database equal to 2.3354. Incremental learning leads to the RMSE equal to 0.3801 for $\varepsilon = 0.01$ and $\tau = 0.25$. Iterative learning leads to the RMSE equal to 0.4243 for $\varepsilon = 0.07$ and $\tau = 0.10$. Thus, in the pres-

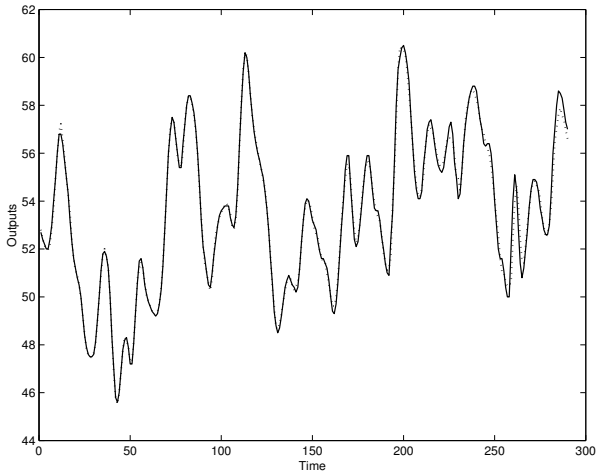


Fig. 3. Output of Box-Jenkins data (a solid line), output of the fuzzy model with learning tolerant of imprecision, the iterative learning with $c = 2$, $\varepsilon = 0.02$, $\tau = 0.02$ (a dotted line).

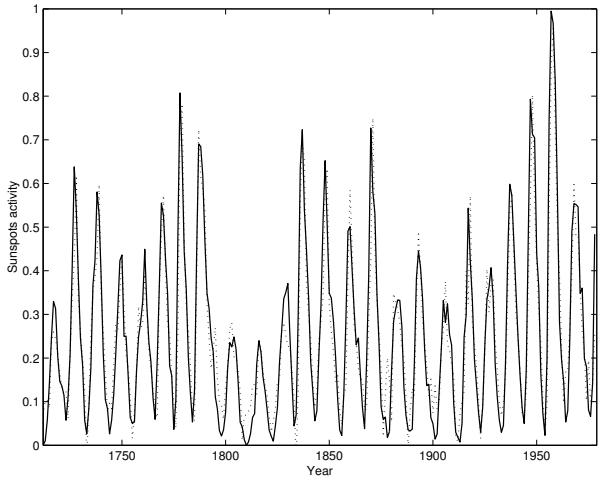


Fig. 4. Sunspot data (a solid line), output of the fuzzy model with learning tolerant of imprecision, the incremental learning with $c = 3$, $\varepsilon = 0.07$, $\tau = 0.1$ (a dotted line).

ence of outliers the proposed learning methods improve the generalization ability of fuzzy systems. If the first two output samples are set to 100, then the zero-tolerant, incremental and iterative learning algorithms lead to the RMSE equal to 4.9753, 0.3867 and 0.4924, respectively.

Interesting questions for future research are as follows: (i) Is it possible to improve the generalization ability of a fuzzy model by its fine tuning using gradient methods based on a ε-insensitive loss function? (ii) Is it true that using the ε-insensitive fuzzy c-means (Łęski, 2001) instead of the FCM clustering improves the generalization ability of a designed fuzzy system?

6. Conclusions

In this work, a new approach to fuzzy modelling with learning tolerant of imprecision is presented. The ε-insensitive loss function is used in this method of learning. Computationally effective numerical methods of designing neuro-fuzzy systems called the incremental and the iterative learning are introduced. These methods establish a connection between fuzzy modelling and statistical learning theory where an easy control of the VC-dimension (system complexity) is permitted. Two numerical examples show the usefulness of the new method in designing fuzzy systems with an improved generalization ability and robustness against outliers when compared with the classical zero-tolerant learning. Learning tolerant of imprecision always leads to a better generalization when compared with classical methods. It is impossible to decide which tolerance learning method is the best one. The incremental learning algorithm has better outlier robustness and for some databases leads to a better generalization with respect to iterative learning, but its computational burden is approximately twice as big. The iterative learning is easier to implement and for some databases leads to a better generalization with respect to incremental learning.

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.

Cauwenberghs G. and Poggio T. (2001): *Incremental and decremental support vector machine learning*. — Proc. IEEE *Neural Information Processing Systems Conference*, Cambridge MA: MIT Press, Vol. 13, pp. 175–181.

Chen J.-Q., Xi Y.-G. and Zhang Z.-J. (1998): *A clustering algorithm for fuzzy model identification*. — Fuzzy Sets Syst., Vol. 98, No. 3, pp. 319–329.

Czogała E. and Łęski J. (2001): *Fuzzy and Neuro-Fuzzy Intelligent Systems*. — Heidelberg: Physica-Verlag, Springer-Verlag Comp.

Gantmacher F.R. (1959): *The Theory of Matrices*. — New York: Chelsea Publ.

Haykin S. (1999): *Neural Networks. A Comprehensive Foundation*. — Upper Saddle River: Prentice-Hall.

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.

Huber P.J. (1981): *Robust Statistics*. — New York: Wiley.

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.

Joachims T. (1999): *Making large-scale support vector machine learning practical*, In: Advances in Kernel Methods—Support Vector Learning (B. Schölkopf, J.C. Burges and A.J. Smola, Eds.). — New York: MIT Press.

Łęski J. (2001): *An ε-insensitive approach to fuzzy clustering*. — Int. J. Appl. Math. Comp. Sci., Vol. 11, No. 4, pp. 993–1007.

Osuna E., Freund R. and Girosi F. (1997): *An improved training algorithm for support vector machines*. — Proc. IEEE Workshop *Neural Networks for Signal Processing*, Breckenridge, Colorado, pp. 276–285.

Pedrycz W. (1984): *An identification algorithm in fuzzy relational systems*. — Fuzzy Sets Syst., Vol. 13, No. 1, pp. 153–167.

Platt J. (1999): *Sequential minimal optimization: A fast algorithm for training support vector machines*, In: Advances in Kernel Methods—Support Vector Learning (B. Schölkopf, J.C. Burges and A.J. Smola, Eds.). — New York: MIT Press.

Rutkowska D. (2001): *Neuro-Fuzzy Architectures and Hybrid Learning*. — Heidelberg: Physica-Verlag, Springer-Verlag Comp.

Rutkowska D. and Hayashi Y. (1999): *Neuro-fuzzy systems approaches*. — Int. J. Adv. Comp. Intell., Vol. 3, No. 3, pp. 177–185.

Rutkowska D. and Nowicki R. (2000): *Implication-based neuro-fuzzy architectures*. — Int. J. Appl. Math. Comp. Sci., Vol. 10, No. 4, pp. 675–701.

Setnes M. (2000): *Supervised fuzzy clustering for rule extraction*. — IEEE Trans. Fuzzy Syst., Vol. 8, No. 4, pp. 416–424.

Sugeno M. and Kang G.T. (1988): *Structure identification of fuzzy model*. — Fuzzy Sets Syst., Vol. 28, No. 1, pp. 15–33.

Takagi H. and Sugeno M. (1985): *Fuzzy identification of systems and its application to modeling and control*. — IEEE Trans. Syst. Man Cybern., Vol. 15, No. 1, pp. 116–132.

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.

Wang L.-X. (1998): *A Course in Fuzzy Systems and Control*. — New York: Prentice-Hall.

Weigend A.S., Huberman B.A. and Rumelhart D.E. (1990): *Predicting the future: A connectionist approach*. — Int. J. Neural Syst., Vol. 1, No. 2, pp. 193–209.

Yen J., Wang L. and Gillespie C.W. (1998): *Improving the interpretability of TSK fuzzy models by combining global learning and local learning*. — IEEE Trans. Fuzzy Syst., Vol. 6, No. 4, pp. 530–537.

Zadeh L.A. (1964): *Fuzzy sets*. — Inf. Contr., Vol. 8, No. 4, pp. 338–353.

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.

Appendix A

The first equation from (37) can be rewritten in the form $\mathbf{X}_e^T\mathbf{D}_e\mathbf{e} = -\frac{\tau}{2}\widetilde{\mathbf{I}}\mathbf{W}$. Thus, for $\tau > 0$ all elements of the error vector cannot be zero. If we define $\mathbf{X}_e^\dagger \triangleq (\mathbf{X}_e^T\mathbf{D}_e\mathbf{X}_e + \tau/2\widetilde{\mathbf{I}})^{-1}\mathbf{X}_e^T\mathbf{D}_e$ and $\mathbf{e}_+^{[k]} \triangleq \mathbf{e}^{[k]} + |\mathbf{e}^{[k]}|$, then using (37) and (38) yields $\mathbf{e}^{[k+1]} = \mathbf{e}^{[k]} + \rho(\mathbf{X}_e\mathbf{X}_e^\dagger - \mathbf{I})\mathbf{e}_+^{[k]}$ and $\widetilde{\mathbf{I}}\widetilde{\mathbf{W}}^{[k+1]} = \mathbf{X}_e^\dagger(\mathbf{b}^{[k]} + \rho\mathbf{e}_+^{[k]}) = \widetilde{\mathbf{W}}^{[k]} + \rho\mathbf{X}^\dagger\mathbf{e}_+^{[k]}$. Substitution of the above results in (36) gives $I^{[k+1]} = I^{[k]} + 2\rho\mathbf{e}^{[k]T}\mathbf{D}_e(\mathbf{X}_e\mathbf{X}_e^\dagger - \mathbf{I})\mathbf{e}_+^{[k]} + \rho^2\mathbf{e}_+^{[k]T}(\mathbf{X}_e\mathbf{X}_e^\dagger - \mathbf{I})^T\mathbf{D}_e(\mathbf{X}_e\mathbf{X}_e^\dagger - \mathbf{I})\mathbf{e}_+^{[k]} + 2\tau\rho\widetilde{\mathbf{W}}^{[k]T}\widetilde{\mathbf{I}}\mathbf{X}_e^\dagger\mathbf{e}_+^{[k]} + \tau\rho^2\mathbf{e}_+^{[k]T}\mathbf{X}_e^{\dagger T}\widetilde{\mathbf{I}}\mathbf{X}_e^\dagger\mathbf{e}_+^{[k]}$. From the first equation of (37) we have $\mathbf{X}_e^T\mathbf{D}_e\mathbf{e}^{[k]} = -\frac{\tau}{2}\widetilde{\mathbf{I}}\widetilde{\mathbf{W}}^{[k]}$.

Using the above and the equality $2\rho\mathbf{e}^{[k]T}\mathbf{D}_e(\mathbf{X}_e\mathbf{X}_e^\dagger - \mathbf{I})\mathbf{e}_+^{[k]} = \rho\mathbf{e}_+^{[k]T}\mathbf{D}_e(\mathbf{X}_e\mathbf{X}_e^\dagger - \mathbf{I})\mathbf{e}_+^{[k]}$, after some simple algebra, we obtain $I^{[k+1]} - I^{[k]} = \rho(\rho-1)\mathbf{e}^{[k]T}\mathbf{D}_e\mathbf{e}_+^{[k]} + \rho^2\mathbf{e}_+^{[k]T}\mathbf{X}_e^{\dagger T}(\mathbf{X}_e^T\mathbf{D}_e\mathbf{X}_e + \tau/2\widetilde{\mathbf{I}})\mathbf{X}_e^\dagger\mathbf{e}_+^{[k]} - 2\rho^2\mathbf{e}_+^{[k]T}\mathbf{D}_e\mathbf{X}_e\mathbf{X}_e^\dagger\mathbf{e}_+^{[k]}$. Since $\mathbf{X}_e^{\dagger T}(\mathbf{X}_e^T\mathbf{D}_e\mathbf{X}_e + \tau/2\widetilde{\mathbf{I}})\mathbf{X}_e^\dagger = \mathbf{D}_e\mathbf{X}_e\mathbf{X}_e^\dagger$, the second and third terms reduce to $-\rho^2\mathbf{e}_+^{[k]T}\mathbf{D}_e\mathbf{X}_e\mathbf{X}_e^\dagger\mathbf{e}_+^{[k]}$.

Thus $I^{[k+1]} - I^{[k]} = \rho(\rho-1)\mathbf{e}^{[k]T}\mathbf{D}_e\mathbf{e}_+^{[k]} - \rho^2\mathbf{e}_+^{[k]T}\mathbf{D}_e\mathbf{X}_e\mathbf{X}_e^\dagger\mathbf{e}_+^{[k]}$. The matrix $\mathbf{D}_e\mathbf{X}_e\mathbf{X}_e^\dagger$ is symmetric and positive semi-definite. As a result, the second term is negative or zero. For $0 < \rho < 1$ the first term is negative or zero. Thus the sequence $I^{[1]}, I^{[2]}, \ldots$ is monotonically decreasing. Convergence requires that $\mathbf{e}_+^{[k]}$ tends to zero (no modification in (37)), while $\mathbf{e}^{[k]}$ is bounded away from zero, due to $\mathbf{X}_e^T\mathbf{D}_e\mathbf{e} = -\tau\widetilde{\mathbf{I}}\mathbf{W}$.