Computer Assisted Mechanics and Engineering Sciences, 7: 39–52, 2000.

Dynamic system approach in sensitivity analysis of neural and fuzzy systems

Martyna Weigl[1,2] and Witold Kosiński[1,3]
[1] *Laboratory of Optical and Computer Methods in Mechanics*
Institute of Fundamental Technological Research, Polish Academy of Sciences
Świętokrzyska 21, 00-049 Warsaw, Poland
[2] *Polska Telefonia Cyfrowa, Al. Jerozolimskie 181, 02-222 Warsaw, Poland*
[3] *Polish–Japanese Institute of Information Technology*
ul. Koszykowa 86, 02-008 Warsaw, Poland

(Received March 2, 1999)

In the paper some results of investigations of two intelligent information systems: a feedforward neural network and an adaptive fuzzy expert system, are presented. The systems can be used for example in approximation and control problems or in diagnostics. The adaptive fuzzy expert system is constructed as a hybrid in which a fuzzy inference system is combined with a neural network. In the learning process for given set of training points an optimal value of the so-called generalized weight vector is searched. The Lapunov theory is used to examine the non-sensitivity of the optimal value of a generalized weight vector to initial conditions and training data. Some necessary and sufficient conditions are formulated in terms of the Hessian matrix of the error function.

Keywords: mapping neural networks, fuzzy inference systems, fuzzy expert systems, training process, optimal value, sensitivity, Lapunov theory, asymptotic stability

1. Introduction

Feedforward neural networks capable to approximate with a given accuracy any continuous real-valued functions defined on a given compact subset A of $\mathbb{R}^n$ and known only in a finite number of points of its domain, are called *mapping neural networks.*

The architecture of a neural network is described by a number of neuron layers and number of neurons on each layer. Each neuron is characterized by its activation multi-parameter function and is related to other neurons in a subsequent layer by means of a multi-parameter affine functions. All those parameters form the so-called *generalized weight vector* Θ and have to be fixed after the so-called *training process* during which the network (i.e. the function realized by the net) is adapted to the known values of the approximated functions in those finite number of points.

There is another type of large intelligent information systems used in the class of problems covered by feedforward neural networks, namely *fuzzy inference systems.* Such systems combined with neural networks can form the so-called adaptive fuzzy systems or the more general class of systems, if constructed with additional module, namely *adaptive fuzzy expert systems.* Sometimes such systems are called *adaptive fuzzy inference* ones, since, in the contrast to the classical expert systems, there is a lack of an explanation facility (cf. [1, 2, 9]).

The structure a fuzzy inference system is similar to a two-wing multi layer feedforward network, in which each node performs a particular function on incoming signals using a set of parameters specific to this node.

Both information systems: feedforward neural networks and adaptive fuzzy inference systems (and the more general: adaptive fuzzy expert systems) can be used for a particular problem is

a learning procedure is build for them according to which parameters of the systems are updated, or better to say: they are adapted to given training data. The latter is called a *set of training pairs*. Each learning procedure has some specific algorithm for finding an optimal set of parameters.

The learning procedure is realized in the form of a the so-called *training process*, which is a problem of minimization of some error function of the system $F(\Theta.\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}, \mathbf{Y}$ represents the whole set of training pairs and Θ describes the set of system parameters, regarded as a vector in multidimensional space.

The training process is a realization of some adaptation algorithm. Adaptation algorithm reflects finding an asymptotically stable equilibrium solution to the following ordinary differential equation for some Θ,

$$\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = -\xi \nabla_{\Theta} F(\Theta; \mathbf{x}, \mathbf{y}), \tag{1}$$

with an initial value $\Theta(0) = \Theta^0$, where $\mathbf{x}, \mathbf{y} \in S$. Here $S \subset \mathbb{R}^{n+1}$ is a set containing all training pairs $(\mathbf{X}, \mathbf{Y}) \subset A \times \mathbb{R}$.

In the sensitivity analysis conditions are formulated for the independence of the equilibrium point of the choice of the initial value $\Theta(0)$ and for the whole set $S \subset \mathbb{R}^{n+1}$ containing the training pairs $(\mathbf{X}, \mathbf{Y}) \subset A \times \mathbb{R}$.

Sensitivity considerations of mathematical models that often have to describe physical systems, are most important in the everyday life of engineers and researchers engaged in modelling. The mathematical models of the systems are idealized, inexactly identified, or themselves are subject to unpredictable changes with time due to influence of other systems. Moreover, the environment or date that have been used to build the models could be not exact or given with some errors. So there is always a discrepancy between the physical reality and the mathematical model.

Such a discrepancy is often due to some parameter deviations. Sensitivity analysis provides the engineers and the researchers with methods for investigating or minimizing the effects of such parameters deviations. This is of particular importance to modern control theory which plays a main role in designing and running high sophisticated systems with prescribed or optimal behaviour on the basis of some mathematical models. Then results are useless in practice if they prove to be very sensitive to parameter changes.

Besides the control theory sensitivity considerations are useful in designing mapping neural networks and in general adaptive systems such as fuzzy inference and expert systems, and in application of gradient descent method, in constructing adaptive algorithms for training processes for such schemes.

Sensitivity analysis has its well developed methods applied to mechanical and electrical engineering ([3]). It is rather not well situated in the computer science nor in the artificial intelligence.

The aim of this paper is to give some fundamental notions and very first results of sensitivity considerations in designing neural networks, fuzzy inference and expert systems.

The present investigations on sensitivity and convergence of learning algorithms, the latter appearing in the constructed expert system, play an important role for the whole designed systems.

2. Fuzzy expert systems versus neural networks

Expert systems are computer consulting programs that perform reasoning using previously established rules for a well-defined and narrow domain. They combine knowledge bases of rules and domain-specific facts with information from users or clients about specific instances of problems in the knowledge domains of the expert systems.

Knowledge supplied to the expert system comes mainly from human experts. It can be in the form of experimental results or measurements, and finally it can come from a survey of the literature related to the domain of the expert system. Knowledge bases in expert systems can be modified. Expert systems are knowledge driven so changes can be made only if the knowledge is being changed.

Expert systems and neural networks are particularly good technologies to integrate because of their complementary natures. However, either or both can be combined with other intelligent subsystems to address deficiencies in current solutions to real-world problems. Hence, in order to mimic an aspect of human reasoning by doing approximate reasoning *fuzzy systems* based on fuzzy set theory [15] and associated techniques should be developed. Fuzzy systems apply techniques that can broaden the usefulness of expert systems, allowing operation in gray areas where precise values may not be known or may not be necessary for drawing conclusions.

The adaptive fuzzy expert system (*AFES*) has to solve some drawbacks of the expert system and to use the main advantages of neural computing and at the same time to give the possibility to deal with imprecise data forming useful conclusions for end users. Adaptive fuzzy expert systems are less precise than conventional systems but are more like our everyday experience as human decision makers. This feature makes knowledge engineering accessible to a wider variety of analysts, end users, and experts.

Constructing an *AFES* one can abandon the help of the human expert in the formation of fuzzy inference rules and membership functions of fuzzy sets involved. The adaptive features of neural networks are used to tune-up (adjust) the shapes of the functions when the training data is given. The neural network component is useful in constructing an optimal set of rules. On the other hand the fuzzy logic makes possible to formalize knowledge in a block of imprecise (fuzzy) rules that mimic more accurate aspects of human reasoning. It is interesting that for the adaptation process the true nature of the information coming in the form of training is irrelevant. However, a uniform distribution of the training sample over the whole domain is relevant for the process and the further use of the system.

A major limitations of expert system approach arises from the fact that experts do not always think in terms of rules. Hence the knowledge acquisition becomes a fundamental limitation of the expert system approach. Another difficulty arises in the area of large system development, i.e. for real-world applications, where the development process becomes difficult to manage. Working with experts and dealing with the complexity of large systems validation and verification of the constructed expert system becomes difficult, if not impossible, as many lines of reasoning must be checked.

Other limitations are related to the fact that expert systems do not automatically benefit from experience with their use of novel examples and thus do not learn from failures.

Adjustment limitations of an expert system to the varying environmental conditions form the other difficulty in fuzzy expert system applications.

Neural networks can be preferable to expert systems when rules are not known, either because the topic is too complex or no human expert is available. If an appropriate (i.e. uniformly distributed in the domain) training data can be generated, the neural network may be able to learn enough information to function as well as, or better than, an expert system. Moreover, modifications are exercised by retraining with updated data set, thus eliminating programming changes and rules reconstruction. The data-driven property of neural networks allows adjustment of changing environments and events. Another feature and advantage of the neural network implementation is the speed of operation after the network is trained.

All above propositions do not prevent the following statement: the adaptation process is equally important for the both systems: feedforward neural networks and adaptive fuzzy systems. Hence the sensitivity analysis devoted to the results of the optimizations problems is necessary.

3. Homogeneous M–DELTA backpropagation learning laws

The architecture of a neural network is described by a number of the neuron layers and number of neurons on each layer. Here we restrict our considerations to one hidden layer. It is worthwhile to mention that from the theoretical point of view each multi layer feedforward network can be designed as one-hidden network.

In the vector notation the output vector z from the network is given by the composition of two, general non–linear activation functions f^I and f^{II} with linear (affine) functions, namely

$$z = f^{II}(\omega^{II} f^I(\omega^I \mathbf{x})), \tag{2}$$

where $\mathbf{x} = (x_0, x_1, \ldots, x_n)$ is the input vector enlarged by the bias component $x_0 = 1$. Each of the activation functions f^L, $L = 1, 2$, depends of a set of the parameters m, δ's. Using a common denotation f_h^L for activation functions of the both layers $L = I$ or II and each neuron $h = j$ or k, we have assumed

$$f_h^L(\lambda) = \frac{m_h^L}{1 + \exp(-\delta_h^L \lambda)} \,. \tag{3}$$

Here λ stands for z_j^I or z_α^{II} input values (activations) in the hidden (I) layer, and the output (II) one, respectively. The control parameters m_h^L of the range of the threshold functions form a vector $\mathbf{M}$ and δ_h^L parameters control the shape of the threshold functions and form a vector $\mathbf{\Delta}$.

Let $u_j^I = f_j^I(z_j^I)$ be output value of j-th neuron in the hidden (I) layer, and $u_\alpha^{II} = f_k^{II}(z_\alpha^{II})$ – output value of neuron α in the output (II) layer. The u's outputs being functions of z's inputs are related through the threshold (activation) functions f_h^L taken from a class of sigmoidal-type functions given by (3). For the p–training pair (pattern) we can write the α-th component of the output vector of the network as:

$$y_\alpha = f_\alpha(\mathbf{x}^p; \mathbf{\Theta}) = u_\alpha^{II} := f_\alpha^{II}(z_\alpha^{II}) \tag{4}$$

with

$$z_\alpha^{II} = \sum_{j=0}^{l} \omega_{\alpha j}^{II} u_j^I, \qquad u_j^I = f_j^I(z_j^I), \qquad z_j^I = \sum_{i=0}^{n} \omega_{ji}^I x_i^p \,,$$

where the zero coordinate of each $\mathbf{x}$ is assumed equal to 1 (and in this way the so-called *bias* can be written under the sum sign), and we have defined $\mathbf{\Theta}$ as the generalized weight vector of the network the components of which consist of three parts. The first part $\mathbf{\Theta}$ is formed of the connection (standard) weights of all neurons of the network, starting with the weight ω_{10} of the first neuron on the input layer and ending with the weight ω_{ml} of the last neuron of the output layer. The next parts of $\mathbf{\Theta}$ are the vectors $\mathbf{\Delta}$ and $\mathbf{M}$, respectively. Let us notice that the both vectors $\mathbf{\Delta}$ and $\mathbf{M}$ form two additional sets of neuron's parameters which are introduced in order to improve the actual sensitivity of the network [4, 8, 11, 12].

In the adaptation algorithm the improvement of the accuracy of the approximation of the function $\mathbf{f}$ is gained by the change of $\mathbf{\Theta}$ in "time" and here is determined by the *error backpropagation* algorithm. This algorithm is based on the minimization of the quadratic error function of the network

$$F(\mathbf{\Theta}, \mathbf{X}, \mathbf{Y}) = \sum_{p=1}^{P} F_p(\mathbf{\Theta}, \mathbf{X}, \mathbf{Y}) \tag{5}$$

in which at each p–presentation the component error function $F_p(\mathbf{\Theta}, \mathbf{X}, \mathbf{Y})$ corresponds to the p-th training pair:

$$F_p(\mathbf{\Theta}, \mathbf{X}, \mathbf{Y}) = \frac{1}{2} \sum_{\alpha=1}^{m} [f_\alpha(\mathbf{x}^p, \mathbf{\Theta}) - y_\alpha^p]^2$$

where $f_\alpha(\mathbf{x}^p, \mathbf{\Theta}) = f_\alpha^{II}(x_1^p, x_2^p, \ldots, x_n^p, \mathbf{\Theta})$ is the α-th component of the output vector obtained in response to the current state of the weight vector $\mathbf{\Theta}$ and $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^P\}$, while $\mathbf{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P\}$. Here a general m-dimensional output is allowed.

After a single presentation, corrections of each weight component are made in the direction of the maximum decrease of $F_p(\Theta)$ given by its negative gradient with respect to Θ. It means that the adaptation algorithm reflects finding an asymptotically stable equilibrium solution to the following ordinary differential equation (compare (1)) for some Θ

$$\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = -\xi \nabla_\Theta F(\Theta; \mathbf{x}, \mathbf{y}), \tag{6}$$

with an initial value $\Theta(0) = \Theta^0$, where $\mathbf{x}, \mathbf{y} \in S$. Here $S \subset \mathbb{R}^{n+1}$ is a set containing all training pairs $(\mathbf{X}, \mathbf{Y}) \subset A \times \mathbb{R}$.

In the numerical treatment the minimization is gained according to so-called learning iterative algorithm:

$$\Theta^1(1) = \Theta^0 - \eta \nabla_\Theta F_1(\Theta^0, \mathbf{X}, \mathbf{Y}), \tag{7}$$

$$\Theta^{s+1}(1) = \Theta^s(P) - \eta \nabla_\Theta F_1(\Theta^s(P), \mathbf{X}, \mathbf{Y}), \tag{8}$$

$$\Theta^s(p+1) = \Theta^s(p) - \eta \nabla_\Theta F_{p+1}(\Theta^s(p), \mathbf{X}, \mathbf{Y}), \tag{9}$$

$$p = 1, 2, \ldots, P-1, \qquad s = 1, 2, \ldots,$$

where Θ^0 is the initial value and the index s correspond to a "time" $t = s\,\Delta\tau$, while $\Delta\tau$ is the time increment and $\eta = \Delta\tau\,\xi$ is called a *learning step.*

4. Adaptive fuzzy expert system AFES

The basic idea of the complex fuzzy expert system is to realize the process of fuzzy reasoning and to express parameters of fuzzy reasoning by connection weights of an neural network and forms of membership functions of fuzzy sets.

In general case a fuzzy expert system *AFES* can be composed of 5 principal elements [8, 11, 14]: *functional consequent part unit*, *fuzzifier of premise parts*, *premise part unit*, *fuzzy rules* and *defuzzifier*. The *fuzzifier* performs a mapping from the crisp input space $A \in \mathbf{R}^n$ to fuzzy sets defined in A, where a fuzzy set is characterized by a membership function $\mu_A : A \longrightarrow [0, 1]$. A *fuzzy rule* consists of a set of linguistic rules in the form: **If** a set of conditions are satisfied, **Then** a set of consequences are satisfied. The *defuzzifier* performs a mapping from fuzzy sets in $\mathbf{R}^m$ to crisp point in $\mathbf{R}^m$.

The proposed approach can either refine the fuzzy *if–then* rules from human experts or automatically derive them from input–output examples. Our adaptive neural network is a two-wing multi layer feedforward network, in which each node performs a particular function on incoming signals using a set of parameters specific to this node.

Let us notice, that if the set X is the range of one of physical variables, (e.g. velocity), then with a fuzzy subset covering $[A_j]_{j=1}^k$ we can combine a set of linguistic variables, by attaching to each of A_i its name, e.g. if $k = 3$, then we can call A_1 – small, A_2 – medium, A_3 – large velocity, for example. In the case of fuzzy rules the condition "x_1 is A_1" can be expressed as follows "x_1 is small velocity" and $\mu_{A_1}(x_1)$ is the level (degree) at which we regard x_1 as a small velocity.

Then in the fuzzifier for each component of input variable x_i a corresponding fuzzy subset covering $[A_j]_{j=1}^k$ is chosen. Let us notice that in general more than one fuzzy subset covering can be defined, especially in the case when knowledge from more than one expert is used.

In the *premise part unit* premise parts of fuzzy rules are constructed and the corresponding crisp weights are determined.

A fuzzy expert system *AFES* maps families of fuzzy sets in to families of functions in the input-output product space $X \times Y$. In the simplest case *AFES* encodes the fuzzy rule (A_L, B_L) in which A_L is a (family of) fuzzy sets defined on the universal sets X of inputs and B_L is a (family of)

functions for Y outputs. A *L-fuzzy rule* (A_L, B_L), where $L = (j_1, \ldots, j_n)$, $A_L = (A_{j_1}, \ldots, A_{j_n})$ can be expressed in the natural language as

$$\textbf{If} \quad x_1 \text{ is } \mathbf{A}_{j_1} \quad \text{and} \quad x_2 \text{ is } \mathbf{A}_{j_2} \ldots \quad \text{and} \quad x_n \text{ is } \mathbf{A}_{j_n} \quad \textbf{Then} \quad y = \mathbf{B}_L(x_1, \ldots, x_n). \tag{10}$$

Firing strength of *L-fuzzy rule* $\mathbf{x}$ is

$$\omega_L(\mathbf{x}) = \prod_{i=1}^{n} \mu_{A_{j_i}}(x_i) \tag{11}$$

where

$$\mu_{A_{j_i}}(x_i) = d^{j_i} \exp\left\{-0.5\left[\left(\frac{(x_i - c^{j_i})}{a^{j_i}}\right)^2\right]^{b^{j_i}}\right\} \tag{12}$$

is a membership function of the set A_{j_i} assumed here as a generalized Gaussian function with 4 parameters [11, 12, 13]. Final output of the *AFES* at $\mathbf{x}$ is

$$z = h(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{R}) = \sum_{L=1}^{M} \omega_L(\mathbf{x}) B_L(\mathbf{x}) \left\{\sum_{L=1}^{M} \omega_L(\mathbf{x})\right\}^{-1}. \tag{13}$$

The constructed for the approximation problem the fuzzy expert system is based on Takagi and Sugeno's fuzzy *if–then* rules forms of which have fuzzy sets involved in premise parts, while consequent part (i. e. output of each rule) is a function of input variable (originally it was an affine function). The final output of the network is the weighted sum of all rule's output. This kind of form of fuzzy rules is one of three types of fuzzy reasoning proposed in the literature. Two other types of rules have fuzzy sets in both: premise and consequent parts.

Then an adaptation procedure for parameters of *AFES* is to design. A typical procedure uses a modified gradient descent method during which the parameters of the system are updated according to given training data.

An adaptation algorithm is built when an error function $F(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{R}, \mathbf{X}, \mathbf{Y})$ depending on the parameters $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{R}$ and the whole set of training pairs $(\mathbf{X}, \mathbf{Y})$ has been chosen. Here

$$\mathbf{a} = \left[a^{j_i}\right]_{j_i=1}^{k_i}, \qquad \mathbf{b} = \left[b^{j_i}\right]_{j_i=1}^{k_i}, \qquad \mathbf{c} = \left[c^{j_i}\right]_{j_i=1}^{k_i}, \qquad \mathbf{d} = \left[d^{j_i}\right]_{j_i=1}^{k_i},$$

are vectors composed of premise part parameters, and

$$\mathbf{R} = \left[R_j^L\right]_{j=0}^{n} \qquad L = 1, 2, \ldots, M,$$

is the vector composed of consequent part parameters. Let $\mathbf{Q} = [\mathbf{B}, \mathbf{R}]$ with $\mathbf{B} = [\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}]$. This vector we will call further a *generalized weight vector.*

The adaptation of the weight vector $\mathbf{Q}$ is gained by the change it in "time" exactly in the same as the generalized weight vector of the neural network $\boldsymbol{\Theta}$ in "time" with the help of a very similar to (1) learning law (compare (6)). We define the error function F as

$$F(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{R}, \mathbf{X}, \mathbf{Y}) = \sum_{q=1}^{P} F_q(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{R}, \mathbf{x}, \mathbf{y}), \tag{14}$$

with

$$F_q(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{R}, \mathbf{x}, \mathbf{y}) = (h(x_1^q, x_2^q, \ldots, x_n^q, \mathbf{Q}) - y^q)^2, \tag{15}$$

where the partial error function $F_q(\mathbf{Q}, \mathbf{x}, \mathbf{y})$ corresponds to the q-th training pair: $(x_1^q, \ldots, x_n^q; y^q)$ with $q = 1, \ldots, P$, and y^q is the target vector, $h(x_1^q, x_2^q, \ldots, x_n^q, \mathbf{Q})$ is the output function from (13),

obtained in the response to the current state of the weight vector $\mathbf{Q}$. The adaptation process is an iterative procedure corresponding to the continuous, ordinary vector differential equation

$$\frac{\mathrm{d}\mathbf{Q}(t)}{\mathrm{d}t} = -\xi \nabla_{\mathbf{Q}} F(\mathbf{Q}; \mathbf{x}, \mathbf{y}), \tag{16}$$

where ξ is responsible for the speed of learning and can depend on the iteration step and $\mathbf{x}, \mathbf{y} \in \mathbf{X}, \mathbf{Y}$) denotes a typical element of the training pairs. To this equation we ought to add the initial conditions: $\mathbf{Q}(0) = \mathbf{Q}^0$. In the numerical treatment the adaptation algorithm relies to calculate succeeding iterative values of the generalized weight vector $\mathbf{Q}^s$, for $s = 1, 2, \ldots$, and $p = 1, 2, \ldots, P$ with initial value $\mathbf{Q}^0 = (\mathbf{B}^0, \mathbf{R}^0)$ according to some learning iterative equations.

Since the error function is quadratic in the vector $\mathbf{Q}_2 = \mathbf{R}$ we are applying the least square method in the process of adaptation of $\mathbf{Q}_2$. Hence we are using two different algorithms for correcting both parts of the generalized weight vector $\mathbf{Q}$.

The first part of weight vector $\mathbf{Q}$, namely $\mathbf{Q}_1 = \mathbf{B}$ is adapted in the course of a non-linear gradient descent method while the second part, namely $\mathbf{Q}_2 = \mathbf{R}$ is adapted according to a recursive least square method (RLS). The first part of iterative equations look like

$$\mathbf{B}^1(1) = \mathbf{B}^0 - \mu \nabla_{\mathbf{B}} F_1(\mathbf{B}^0, \mathbf{R}^0, \mathbf{x}, y),$$

$$\mathbf{B}^{s+1}(1) = \mathbf{B}^s(P) - \mu \nabla_{\mathbf{B}} F_1(\mathbf{B}^s(P), \mathbf{R}^s(P), \mathbf{x}, y), \quad \text{for } s = 1, 2, \ldots, \tag{17}$$

$$\mathbf{B}^s(p+1) = \mathbf{B}^s(p) - \mu \nabla_{\mathbf{B}} F_{p+1}(\mathbf{B}^s(p), \mathbf{R}^{s-1}(P), \mathbf{x}, y), \tag{18}$$

$$\text{for } p = 1, 2, \ldots, P-1, \quad \text{and} \quad s = 2, 3, \ldots,$$

$$\mathbf{B}^1(p+1) = \mathbf{B}^1(p) - \mu \nabla_{\mathbf{B}} F_{p+1}(\mathbf{B}^1(p), \mathbf{R}^0, \mathbf{x}, y), \tag{19}$$

$$\text{for } p = 1, 2, \ldots, P-1, \quad \text{and} \quad s = 1, 2, \ldots.$$

Here the index s corresponds to a "time instant" $t = s\,\Delta\tau$, while $\mu = \Delta\tau\xi$ is the learning step.

5. Sensitivity analysis of adaptation algorithms

Sensitivity theory can be treated as a section of a general system theory, taking into account parameter variations as inputs instead of signals. It is useful to subdive sensitivity theory into two categories: *sensitivity analysis and synthesis.* Sensitivity analysis provides the basic methods to study the sensitivity of a system under consideration to parameter variation. On the other hand, according to [3] sensitivity synthesis is defined as the design of dynamic systems, especially feedback systems, with due regard to sensitivity to parameter variation. In this paper we will be concerned mainly with the methods of sensitivity analysis.

To examine sensitivity of the training (adaptation) processes for both intelligent systems discussed in the previous two sections we will use a common denotation for their training pairs and the generalized weight vectors $\boldsymbol{\Theta}$ and $\mathbf{Q}$.

For discussing the sensitivity of a system with respect to different parameters, input data from the training set, we introduce an example of sensitivity measure.

Let TRE be the training set defined by :

$$TRE = \{(\mathbf{x}^q, \mathbf{y}^q) \mid \mathbf{y}^q = g(\mathbf{x}^q), \quad q = 1, 2, 3, \ldots, P\}, \tag{20}$$

where P is the number of elements of training set. Let the output (transition) of an intelligent system (i.e. a neural network or a fuzzy system) will be characterized by a quantity

$$\mathbf{z} = \mathbf{f}(\mathbf{x}, \boldsymbol{\Theta}),$$

with $\mathbf{x} \in \mathbb{R}^m$, and called a system or output function, which among a dependence on the input vector $\mathbf{x} \in \mathbb{R}^n$, is a function of the generalized weight vector $\boldsymbol{\Theta}$. We may introduce the sensitivity of the output with respect to the input vector $\mathbf{x}$ represented by the set of training pairs TRE. This

is not typical in the classical sensitivity theory (cf. [3]). However, in the process of construction and adaptation of a given structure of a neural network to the training set *TRE*, one wants to check which of output components are relevant to the *TRE* and which are redundant. On the other hand in the adaptation process we can follow the classical approach [3], where sensitivity functions are given in terms of the partial derivatives of the system function $\mathbf{z} = \mathbf{f}(\mathbf{x}, \Theta)$ with respect to Θ. In such a process we are interested in the proper choice of an optimal value of Θ and training pairs from *TRE* are regarded as parameters. Then we define the next notion.

Definition 1. *Sensitivity of the system output* $\mathbf{f}(\mathbf{x}, \Theta)$ *with respect to the initial choice of the weight vector is defined as*

$$\mathbf{C} = \mathbf{C}(\mathbf{x}, \Theta(t), \Theta^0) := \nabla_{\Theta^0} \mathbf{f}(\mathbf{x}, \Theta(t)). \tag{21}$$

This sensitivity matrix will appear in the restricted form in the further analysis.

As it was shown in the previous sections for both intelligent information systems characterized by the error function system $F(\Theta; \mathbf{x}, \mathbf{y})$ the optimal value of the weight vector Θ is determined with the help of an iterative adaptation algorithm corresponding to a discrete version of a continuous process that deals with finding an equilibrium solution of a vector differential equation

$$\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = -\xi \nabla_\Theta F(\Theta; \mathbf{x}, \mathbf{y}), \tag{22}$$

with an initial value $\Theta(0) = \Theta^0$ and $\mathbf{x}, \mathbf{y} \in \mathcal{D}$, where $\mathcal{D}$ contains *TRE*.

Let us notice that the minimization of the error function F by changing the vector Θ in "time" t is a kernel of the learning process. If the minimal value of the error function is reached at some value Θ^* then the gradient of F with respect to Θ should vanish at this point. If this minimal value is the same for all elements $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, where set $\mathcal{D}$ contains the training set *TRE*, then we can end the optimization process and at the same time the adaptation of the value of the weight vector Θ, i.e. the learning process. From the point of view of the differential equation (6) that solution will be constant. Now coming the "time" t that make the order in the learning process, we can see that if there exist t^* such that $\Theta(t^*) = \Theta^*$ that

$$\nabla_\Theta F(\Theta^*; \mathbf{x}, \mathbf{y}) = 0 \quad \text{then} \quad \frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = 0 \quad \text{for} \quad t \geq t^* \,. \tag{23}$$

Let us use the following denotation for the right hand side of the system (6).

$$\mathbf{U}(\Theta; \mathbf{x}, \mathbf{y}) = -\xi \nabla_\Theta F(\Theta; \mathbf{x}, \mathbf{y}). \tag{24}$$

Now, according to the classical concepts form the theory of dynamical systems we can introduce the following

Definition 2. *Let the following system of the differential equations*

$$\frac{\mathrm{d}\Theta}{\mathrm{d}t} = \mathbf{U}(\Theta; \mathbf{x}, \mathbf{y}), \qquad \Theta(0) = \Theta^0, \qquad (\mathbf{x}, \mathbf{y}) \in \mathcal{D} \subset \mathbb{R}^{(n+m)} \tag{25}$$

is given. A point Θ^* *is called the point of equilibrium of the system (25), if* $\Theta(t) \equiv \Theta^*$ *is the solution of this system, where*

$$\mathbf{U}(\Theta^*; \mathbf{x}, \mathbf{y}) = \mathbf{0}.$$

In the adaptation process the interesting point of equilibrium corresponds to the optimal choice of the (generalized) weight vector Θ, for which the function F reaches the minimum for *all* $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$.

To examine the sensitivity of the adaptation algorithm with respect to:

- the choice of the initial parameters Θ^0 of the system,
- to choice of the elements of the training set TRE,

we can formulate the following common requirements:

1. There exists a point of equilibrium, as a limit value of the solution $\Theta(t)$ for t tending to infinity, for each choice of the initial value Θ^0 from some non-empty set $\mathcal{O}^*$.
2. There exists a point of equilibrium, independent of elements $(\mathbf{x}, \mathbf{y})$ from the set $\mathcal{D}$.
3. The value of the equilibrium point is independent of the initial value Θ^0 and of elements $(\mathbf{x}, \mathbf{y})$ from $\mathcal{D}$.

The Lapunov theory [6] examines the necessary and sufficient conditions for the stability and asymptotic stability of nonlinear system differential equations. There exist theorems concerning the continuous dependency of solutions on the initial conditions and initial value of parameters. For a clarity we write

$$\frac{\mathrm{d}\Theta}{\mathrm{d}t} = \mathbf{U}(\Theta; \mathbf{Z}), \qquad \Theta \in \mathbb{R}^Q, \qquad \mathbf{Z} = (\mathbf{x}, \mathbf{y}) \in \mathcal{D} \subset \mathbb{R}^{(n+m)} \tag{26}$$

instead of (25). Following the standard approach and developing the right hand side of Eq. (26) in the vicinity of Θ_0 :

$$\mathbf{U}(\Theta; \mathbf{Z}) = \mathbf{U}(\Theta_0; \mathbf{Z}) + \nabla_\Theta \mathbf{U}(\Theta_0; \mathbf{Z})(\Theta - \Theta_0) + \mathbf{R}_2(\Theta; \mathbf{Z}), \tag{27}$$

we obtain the corresponding equation in the first approximation, where the third element is small of the second order

$$\mathbf{R}_2 = O(\mid \Theta - \Theta_0 \mid^2). \tag{28}$$

Due to $\mathbf{U}(\Theta; \mathbf{x}, \mathbf{y}) = -\xi \nabla_\Theta F(\Theta; \mathbf{x}, \mathbf{y})$, the square matrix $\nabla_\Theta \mathbf{U}(\Theta_0; \mathbf{Z})$ is proportional to the Hessian $\mathbf{H}$ of the error function F at the point Θ_0,

$$\nabla_\Theta \mathbf{U}(\Theta_0; \mathbf{Z}) = -\xi \mathbf{H}(\Theta_0; \mathbf{Z}), \qquad \mathbf{H}(\Theta; \mathbf{Z}) := \nabla_\Theta \nabla_\Theta F(\Theta; \mathbf{Z}). \tag{29}$$

Notice, that if $\Theta_0 = \Theta^*$, the first element in (27) vanishes. If Θ_0 is not a point of equilibrium, then $\mathbf{U}(\Theta_0; \mathbf{Z}) \neq 0$ and in the linearized case we will search for a solution of the nonhomogeneous system of linear equations.

Notice, that the Hessian is a symmetric matrix and all its eigenvalues are real. Hence the condition for the negative Hessian $\mathbf{H}$ to be a stability matrix can be formulated as the requirement for $\mathbf{H}$ to be *a positive matrix.*

The basic result of the qualitative theorem tells that the stability properties of the nonlinear system is the same as the linearized one, when the nonlinear term $\mathbf{R}_2$ is bounded. Hence assuming that on the domain $\mathcal{O}^* \times \mathcal{D}$ the nonlinear part $\mathbf{R}_2$ is bounded we can formulate the following:

Theorem 1. *The sufficient condition for the point of equilibrium Θ^* of the nonlinear system (6) for each $\mathbf{Z} = (\mathbf{x}, \mathbf{y}) \in \mathcal{D} \subset \mathbb{R}^Q$,*

$$\frac{\mathrm{d}\Theta}{\mathrm{d}t} = -\xi \nabla_\Theta F(\Theta; \mathbf{Z}), \tag{30}$$

to be asymptotically stable is that the Hessian $\mathbf{H}$ is a positive matrix. And moreover

- *The point of equilibrium is* not sensitive to a choice of the initial value, *i.e. the function $\mathbf{w}(t) := \nabla_{\Theta^0} \Theta(t)$ tends to the zero matrix for the increasing to infinity "time" (i.e. when t tends to ∞), provided the Hessian of the error function is positive.*

- *The point of equilibrium is* not sensitive to a choice of the training vectors, *i.e. the function* $\mathbf{W}(t) := \nabla_{\mathbf{Z}}\boldsymbol{\Theta}(t)$ *tends to the zero matrix for the increasing to infinity time, provided the Hessian of the error function is positive. and the mixed gradient of the error function* $\mathbf{M} := \nabla_{\mathbf{Z}}\nabla_{\boldsymbol{\Theta}}F$ *vanishes at the point of equilibrium for* $\mathbf{Z}$,

$$\mathbf{M}(\boldsymbol{\Theta}^*;\mathbf{Z}) := \nabla_{\mathbf{Z}}\nabla_{\boldsymbol{\Theta}}F(\boldsymbol{\Theta}^*;\mathbf{Z}) = \mathbf{0}, \qquad \text{for } \mathbf{Z} \in \mathcal{D}. \tag{31}$$

□

6. Sensitivity analysis of the shape of membership functions

In the case of fuzzy inference and expert systems the choice of the initial shape of the membership functions of fuzzy numbers appearing in premise parts of fuzzy rules can influence the final adapted form. Let us try to formulate the conditions of its sensitivity.

If in a fuzzy system the weight vector $\boldsymbol{\Theta}$ is adapted in the course of solving a differential equation of the type (25) with the output function $\mathbf{f}(\mathbf{x},\boldsymbol{\Theta})$, then we can define a corresponding sensitivity matrix.

Definition 3. *Sensitivity of the system output* $\mathbf{f}(\mathbf{x},\boldsymbol{\Theta})$ *to the choice of the initial shape of the membership function, i.e. to the choice of the initial values of the parameters* $\boldsymbol{\Theta}_1^0$, *is the matrix function*

$$\mathbf{C} = \mathbf{C}(\mathbf{x},\boldsymbol{\Theta}(t),\boldsymbol{\Theta}^0) := \nabla_{\boldsymbol{\Theta}_1}\mathbf{f}(\mathbf{x},\boldsymbol{\Theta}(t))\nabla_{\boldsymbol{\Theta}_1^0}\boldsymbol{\Theta}_1(t) \tag{32}$$

where

$$(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{R}) = (\boldsymbol{\Theta}_1,\boldsymbol{\Theta}_2).$$

It is a submatrix of the matrix $\mathbf{C}$ defined by (21). From the classical results of qualitative analysis of dynamical system governed by differential equations follows that the gradient of solution of (25) with respect to the initial condition, i.e. the vector $\mathbf{w}(t) = \nabla_{\boldsymbol{\Theta}_1^0}\boldsymbol{\Theta}_1(t)$ that describes the sensitivity of actual values of the membership function parameters $\boldsymbol{\Theta}_1(t)$ to its initial value $\boldsymbol{\Theta}_1^0$, satisfies the equation:

$$\frac{\mathrm{d}\mathbf{w}(t)}{\mathrm{d}t} = -\xi\,\mathbf{H}(\boldsymbol{\Theta}(t);\mathbf{Z})\mathbf{w}(t) \tag{33}$$

with the initial condition $\mathbf{w}(0) = \mathbf{1}$. Moreover, the solution of (33) tends to zero with $t \to \infty$, if the Hessian $\mathbf{H}$ is positive defined.

Notice, that from (32) we obtain

$$\mathbf{C}(t) = \nabla_{\boldsymbol{\Theta}_1} f(\mathbf{x},\boldsymbol{\Theta})\,\mathbf{w}(t). \tag{34}$$

Now we may formulate

Corollary 1. *If the assumptions of Theorem 1 are satisfied and the gradient* $\nabla_{\boldsymbol{\Theta}_1^0} f(\mathbf{x},\boldsymbol{\Theta})$ *is bounded then at a equilibrium point (i.e. for the optimum value of weight of vector* $\boldsymbol{\Theta}^*$ *) the output of the system is* not sensitive to the initial shape of membership function.

Corollary 2. *If at the equilibrium point the gradient of the system output* $\nabla_{\boldsymbol{\Theta}_1^0} f(\mathbf{x},\boldsymbol{\Theta}^*)$ *vanishes, then the same happens with the sensitivity matrix* $\mathbf{C}(\mathbf{x},\boldsymbol{\Theta}^*,\boldsymbol{\Theta}^0)$.

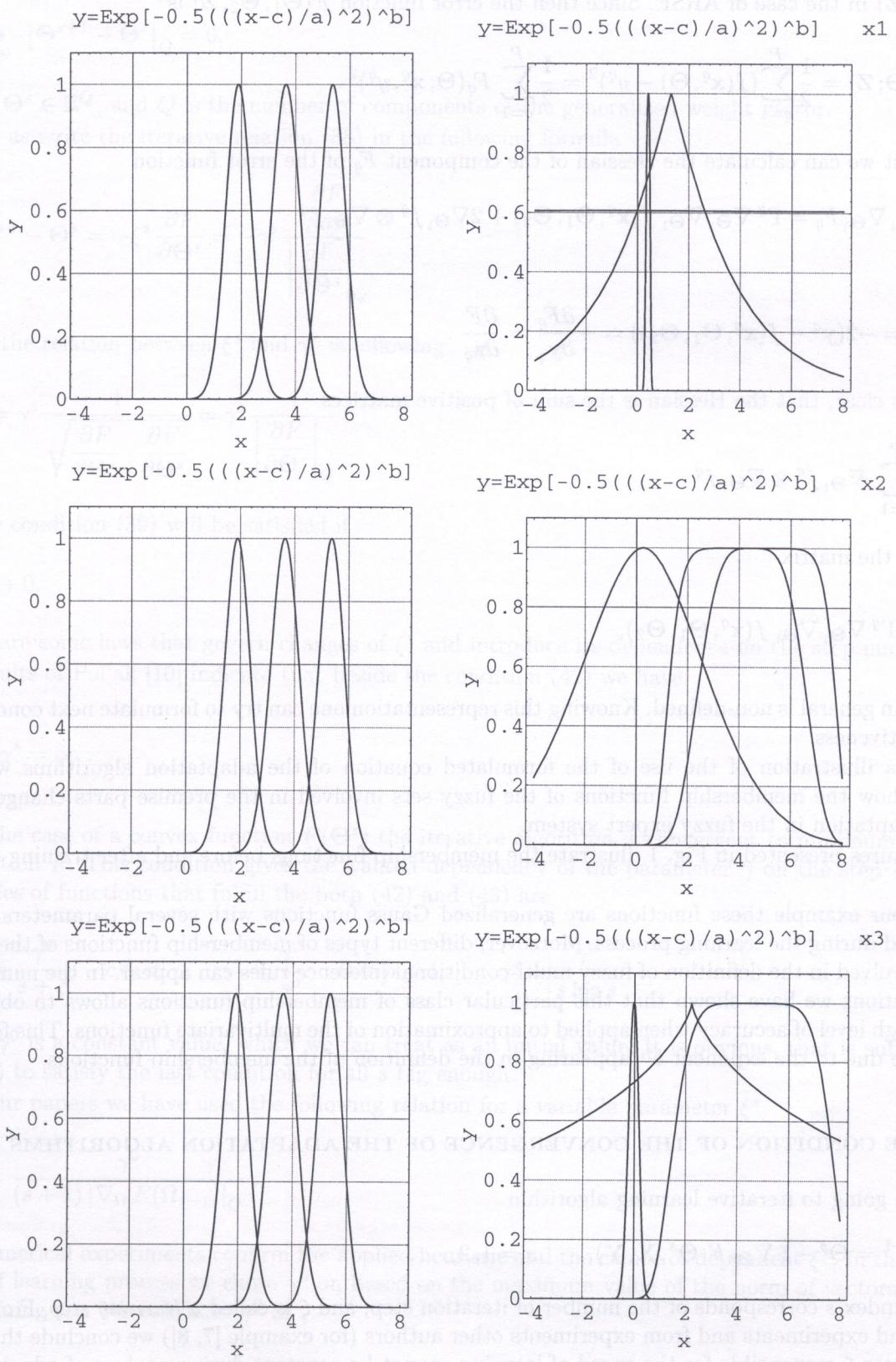

Fig. 1. Membership functions before and after learning, x, y, z – coordinates

We will close this section with a short discussion concerning the representation of the Hessian $\mathbf{H}(\boldsymbol{\Theta};\mathbf{Z})$ in the case of ARSE. Since then the error function $F(\boldsymbol{\Theta}_1,\boldsymbol{\Theta}_2,\mathbf{Z})$ is

$$F(\boldsymbol{\Theta};\mathbf{Z}) = \frac{1}{2}\sum_{q=1}^{P}(f(\mathbf{x}^q,\boldsymbol{\Theta}) - y^q)^2 = \frac{1}{2}\sum_{q=1}^{P} F_q(\boldsymbol{\Theta};\mathbf{x}^q,y^q)^2\,. \tag{35}$$

First we can calculate the Hessian of the component F_q of the error function

$$\nabla_{\boldsymbol{\Theta}_1}\nabla_{\boldsymbol{\Theta}_1}F_q = \Gamma^q\,\nabla_{\boldsymbol{\Theta}_1}\nabla_{\boldsymbol{\Theta}_1}f(\mathbf{x}^q,\boldsymbol{\Theta}_1,\boldsymbol{\Theta}_2) + 2\nabla_{\boldsymbol{\Theta}_1}f^q\otimes\nabla_{\boldsymbol{\Theta}_1}f^q \tag{36}$$

where

$$\Gamma^q = -2(\mathbf{y}^q - f(\mathbf{x}^q,\boldsymbol{\Theta}_1,\boldsymbol{\Theta}_2)) = -\frac{\partial F_q}{\partial \mathbf{y}_q} = \frac{\partial F}{\partial \mathbf{z}_q}\,. \tag{37}$$

It is clear, that the Hessian is the sum of positive matrices

$$2\sum_{q=1}^{P}\nabla_{\boldsymbol{\Theta}_1}f^q\otimes\nabla_{\boldsymbol{\Theta}_1}f^q$$

and of the matrix

$$\sum_{q=1}^{P}\Gamma^q\,\nabla_{\boldsymbol{\Theta}_1}\nabla_{\boldsymbol{\Theta}_1}f(\mathbf{x}^q,\boldsymbol{\Theta}_1,\boldsymbol{\Theta}_2),$$

which in general is non-defined. Knowing this representation one can try to formulate next conditions of positiveness.

As a illustration of the use of the formulated equation of the adaptation algorithms we will show, how the membership functions of the fuzzy sets involved in the premise parts change after the adaptation in the fuzzy expert system.

Pictures presented in Fig. 1 illustrate the membership functions before and after training procedure.

In our example these functions are generalized Gauss functions with several parameters to be adapted during the learning process. Moreover, different types of membership functions of the fuzzy sets involved in the definition of fuzzy multi-conditional inference rules can appear. In the numerical applications we have shown that this particular class of membership functions allows to obtain a very high level of accuracy when applied to approximation of the multivariate functions. This feature may be due to the exponent b^{j_i} appearing in the definition of the membership functions.

7. The condition of the convergence of the adaptation algorithms

We are going to iterative learning algorithm

$$\boldsymbol{\Theta}^{s+1} = \boldsymbol{\Theta}^s - \xi\,\nabla_{\boldsymbol{\Theta}^s}F(\boldsymbol{\Theta}^s,\mathbf{X},\mathbf{Y}), \qquad s = 1,2,\ldots, \tag{38}$$

where index s corresponds of the number of iteration step, and ξ is called *a learning step*. From our tests and experiments and from experiments other authors (for example [7, 8]) we conclude that the parameter ξ responsible for the speed of learning cannot be constant during process of adaptation. In the first stage of learning, when we are far from optimal solution ξ ought to be large, so that the process reaching the minimum of the error function is quick enough. Close to the optimal solution ξ should decrease in order to omit large oscillations and to speed up the process of learning.

The necessary condition of convergence of the learning algorithm is the following relation

$$\lim_{s\to\infty} \left|\Theta^{s+1} - \Theta^{s}\right|_{Q} = 0, \tag{39}$$

where $\Theta^s \in \mathbb{R}^Q$, and Q is the number of components of the generalized weight vector.

Let us write the iterative relation (38) in the following formula

$$\Theta^{s+1} - \Theta^{s} = -\xi^s \frac{\partial F}{\partial \Theta^s} = -\gamma^s \frac{\dfrac{\partial F}{\partial \Theta^s}}{\left|\dfrac{\partial F}{\partial \Theta^s}\right|_{Q}} \tag{40}$$

where the relation between ξ^s and γ^s is following

$$\xi^s = \gamma^s \frac{1}{\sqrt{\dfrac{\partial F}{\partial \Theta^s} \cdot \dfrac{\partial F}{\partial \Theta^s}}} = \gamma^s \frac{1}{\left|\dfrac{\partial F}{\partial \Theta^s}\right|_{Q}} . \tag{41}$$

The condition (39) will be satisfied if

$$\gamma^s \to 0. \tag{42}$$

There are some laws that govern changes of ξ^s and introduce its dependence on the step number.

Results of Pol'ak [10] indicate that beside the condition (42) we have

$$\sum_{s=0}^{\infty} \gamma^s = \infty\,, \tag{43}$$

In the case of a convex function $F(\Theta^s)$ the iterative algorithm is convergent to minimum value of function F. This condition gives the wanted dependency of the parameter γ on the step s. The examples of functions that fulfill the both (42) and (43) are

$$\gamma^s = \frac{\gamma^0}{s+c}, \quad c \in \mathbb{R}; \qquad \gamma^s = \frac{\gamma^0}{s^\rho}, \quad 0 < \rho \leq 1; \qquad \gamma^s = \frac{\gamma^0}{s \log s}\,, \tag{44}$$

where γ^0 is a constant value, which we can treat as an initial value. It is obvious, that is sufficient for (44) to satisfy the last condition for all s big enough.

In our papers we have used the following relation for a variable parameter ξ^s

$$\xi^s = \frac{\gamma^0}{(s+1)\left|\nabla_\Omega F(\Omega_{s-1})\right|_{Q}} . \tag{45}$$

The numerical experiments confirm the applied heuristic and the choice of dependent ξ^s. In the first stage of learning process we chose γ^0 on based on the maximum value of the norm of vectors from the training set. We calculate γ^0 as

$$\gamma^0 = \frac{1}{\max\limits_{p=1,2,\ldots,P} \left(\sum\limits_{i=0}^{n} (x_i^p)^2\right)}\,, \qquad x_0^p = 1. \tag{46}$$

8. Conclusions

Derived and formulated in Sec. 6 conditions of sensitivity and nonsensitivity are formulated in the terms of matrices of partial differential equations of the error function, and hence they are local conditions. It means that they can be treated as some hints for sensitivity of the actual iterative adaptation laws. However, having those conditions we can try to formulate their iterative counterparts.

In our opinion the crucial characterization of the insensitivity of the optimal value of the generalized weight vector, at least in the continuous case, is the positiveness of the Hessian matrix of the error function. Hence, in either case, one should control the sign of the eigenvalues of the Hessian during the iteration process, in order to stay on the secure side of the optimization process.

As far as the condition of the convergence of the adaptation algorithms in the paper [4], the authors make some simulation with proposed above formulae for the parameter ξ responsible for the speed of learning. Further numerical simulation are made in the recent paper [5]. The more detailed discussion will be made in the doctoral thesis of Mr. Gołąbek, which is under preparation.

Acknowledgements

This paper has been prepared in the course of studies supported by the grant No. 8 T11C 011 12 from the Polish State Committee for Scientific Research (KBN).

References

[1] L. Bolc and M.J. Coombos, eds. *Expert System Applications.* Springer-Verlag, 1988.

[2] J. Buckley, Y. Hayashi, and E. Czogała. On the equivalence of neural nets and fuzzy expert systems. *Fuzzy Sets and Systems, North-Holland*, **53**: 129–134, 1993.

[3] P.M. Frank. *Introduction to System Sensitivity Theory.* Academic Press, New York, 1978.

[4] P. Gołąbek, W. Kosiński and M. Weigl. Adaptation of learning rate via adaptation of weight vector in modified M-Delta networks. In: P.S. Szczepaniak (ed.), *Computational Intelligence and Applications*, (*Studies in Fuzziness and Soft Computing*, Vol. **23**), 156–163. Physica-Verlag, c/o Springer-Verlag, 1999.

[5] P. Gołąbek, W. Kosiński, A. Januszewska and M. Kubacka. Numerical experiments with an NM-Delta adaptation algorithm for neural network. Under preparation, 1999.

[6] Ph. Hartman. *Ordinary Differential Equations.* J. Wiley, New York–London–Sydney, 1964.

[7] J. Jang. *ANFIS: Adaptive-Network-Based Fuzzy Inference System.* Technical report, University of California, Berkeley, 1993.

[8] W. Kosiński and M. Weigl. General mapping approximation problems solving by neural networks and fuzzy inference systems. *Systems Analysis Modelling Simulation*, **30**(1): 1998, 11–28.

[9] L. Medsker. *Hybrid Neural Network and Expert Systems.* Kluwer Academic Publishers, Boston–Dordrecht–London, 1994.

[10] B. Pol'ak. *Introduction to Optimization* (in Russian). Nauka, Moscow, 1983.

[11] M. Weigl. *Neural Networks and Fuzzy Inference Systems in Approximation Problems* (in Polish). Ph.D. Thesis, June 1995.

[12] M. Weigl and W. Kosiński. Fuzzy inference system and modified back—propagation network in approximation problems. In: *Proceedings of the III–rd International Symposium on Intelligent Information Systems, June 1994, Wigry n. Suwałki*, 427–442. IPI PAN, Warszawa, 1994.

[13] M. Weigl and W. Kosiński. A neural network system for approximation and its implementation. In: *Proceedings of the First National Conference on Neural Networks and their Applications, April 1994, Kule n. Częstochowa*, 485–491. Politechnika Częstochowska, 1994.

[14] M. Weigl and W. Kosiński. Fuzzy reasoning in adaptive expert systems for approximation problems. In: *Proceedings of the 3-th Zittau Fuzzy – Colloquy Zittau, September 5-6, 1995*, 163–174. Wissenschaftliche Berichte, Heft 41, 1995.

[15] L. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, **11**: 199–226, 1983.