FEEDFORWARD NETWORKS WITH FUZZY SIGNALS [1]

R. Bělohlávek

Institute for Research and Applications of Fuzzy Modeling, University of Ostrava
Bráfova 7, 701 03 Ostrava, Czech Republic, e-mail: belohlav@osu.cz
and
Department of Computer Science, Technical University of Ostrava
tř. 17. listopadu, 708 33 Ostrava-Poruba, Czech Republic

Abstract. The paper discusses feedforward neural networks with fuzzy signals. We analyze the feedforward phase and show some properties of the output function. Then we present a backpropagation like adaptation algorithm for crisp weights, thresholds and neuron slopes of the multilayer network with sigmoidal transfer functions. We provide theoretical justification for the adaptation formulas. The results are of general nature and together with the presented approach can be used for other types of feedforward networks. Proposed and discussed are also applications of the presented feedforward networks.

Key words: feedforward neural networks, fuzzy sets, extension principle, adaptation, backpropagation.

1 Introduction

Fuzzy methods and neural networks constitute the essential part of soft-computing, a discipline aiming at methods capable of dealing with complex humanistic systems. Among the most important features of neural networks we recognize first of all the massively parallel architecture and dynamics which is inspired by the structure of human brain, the adaptation capabilities, and the fault tolerance. On the other hand, fuzzy sets represent an useful model of vagueness, a phenomenon playing crucial role in the way humans regard the world. The significant properties of fuzzy methods are the efficient processing of the modeled indeterminacy, especially the capability of modeling and further processing of natural language semantics. This results in human-friendly models—the models, inputs and outputs can be expressed or interpreted on the level of natural language or have other clearly understandable meaning. Both neural networks and fuzzy methods are closely related with human cognition, however, on different levels. Neural networks concern the microlevel—the level of brain. Fuzzy methods concern the macrolevel—the level of mental phenomena. The levels (micro and macro) are strongly connected. The macrolevel seems to be "implemented" in the microlevel. The analysis of this combination of the levels is fundamental for our understanding of human cognition. The study of the analogous combination of the soft-computing-methodics counterparts of the levels, i.e. neural networks and fuzzy methods, is therefore worth pursuing.

Not surprisingly, neural and fuzzy models can be combined in many ways. The resulting models can be built up of separate neural and fuzzy modules performing relatively independent tasks, they can be based on neural models supplied by some features of fuzzy models (esp. processing of indeterminate information), or conversely, based on fuzzy models supplied by features of neural

[1]Supported by grant No. 201/96/0985 of the GAČR and partially by IGS of University of Ostrava, No. 031/97, 029/98.

networks (esp. adaptability). An overview of various kinds of these models, called neuro-fuzzy or fuzzy-neuro models, can be found e.g. in [4, 7, 11, 16].

The combinations of neuro and fuzzy models presented in literature are often *ad hoc* approaches without correct mathematical treatment. To bring some order into the development it is necessary to delineate and to pursue some paradigms of the combination. A useful paradigm is represented by Zadeh's *extension principle* (EP) [9, 12, 14]. The combination based on EP has been firstly presented probably in [4]. Our paper deals with this approach and is organized as follows.

In Section 2, we propose the architecture and deal with the feedforward phase of our model. In Section 3, we propose an adaptation procedure and derive the adaptation formulas. Possible applications are discussed in Section 4. An example demonstrating the learning and interpolation capabilities is presented in Section 5. Section 6 briefly summarizes the results.

2 Architecture and feedforward phase

By a *feedforward (neural) network* [1] it is usually understood a networked computational scheme composed of several simple computational units which are connected in such a way that the graph of information flow contains no loops (the information is fed forward). The simple units process information in a way analogous to that of biological neurons. The adjective "neural" is due to the analogy with biological neural networks. The feedforward networks take their inputs, process them in the feedforward phase and return the outputs. The inputs to the network are usually real numbers. In what follows we discuss the case where the inputs and the outputs are not real numbers but fuzzy sets in the real line. The reason and advantages of having fuzzy sets instead of crisp real numbers are discussed in Section 4.

There are several particular types of feedforward networks, see e.g. [1]. Probably the most popular one is the multilayer feedforward network with sigmoidal transfer functions called also backpropagation net because of its adaptation procedure [17]. This network will be a subject of our investigation. However, our results which concern the general case of feedforward networks will be formulated as general as necessary for application to other types of feedforward networks.

We now briefly describe the multilayer feedforward neural net. We suppose the j–th neuron of the l–th layer to have the sigmoidal transfer function $y_j^{(l)} = \frac{1}{1+e^{-\lambda_j^{(l)} z_j^{(l)}}}$, where $\lambda_j^{(l)}$ is the slope of $y_j^{(l)}$. The input $z_j^{(l)}$ to the neuron is the thresholded weighted sum of the outputs of the preceding layer, i.e. $z_j^{(l)} = \sum_{i=1}^{m_{l-1}} w_{ij}^{(l)} y_i^{(l-1)} - \theta_j^{(l)}$. Here $w_{ij}^{(l)}$ denotes the weight of the connection leading from the i–th neuron of the $(l-1)$–th layer to the j–th neuron of the l–th layer, and $\theta_j^{(l)}$ is the threshold. A part of the network is depicted in Fig. 1. Let m_l and r denote the number of neurons in the l–th layer and the number of layers in the network, respectively. The inputs to the network are formally the outputs of the 0–th layer. Such a network performs a mapping $F : \mathcal{R}^{m_0} \mapsto (0,1)^{m_r}$. A suitable transformation can guarantee that the outputs are from $\mathcal{R}^{m_r}$.

From now on we suppose the inputs to the neural network to be fuzzy sets in $\mathcal{R}$ instead of real numbers. One can then use the EP to extend the operations performed by the neural network to fuzzy sets. We apply the EP to each operation performed by the net. To be able to cope effectively with EP we restrict ourselves to the class $\mathcal{F}_{CI}(\mathcal{R})$ of normal convex fuzzy sets in $\mathcal{R}$ with compact α–cuts for $\alpha \in (0,1]$. Recall that a fuzzy set A is normal if there is an x with $A(x) = 1$; the α–cut of A is the set ${}^{\alpha}A = \{x; A(x) \geq \alpha\}$; A is convex if each of its α–cuts is convex, i.e. an interval in our case; each fuzzy set x may be represented by a collection $\{{}^{\alpha}x; \alpha \in (0,1]\}$ of its α-cuts. Under

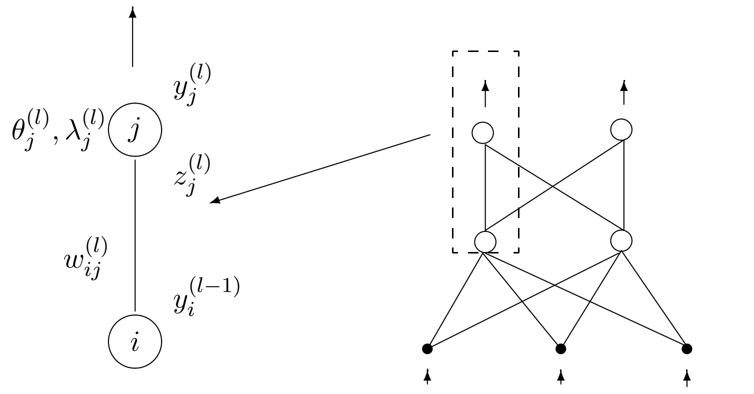

Figure 1: Part of the underlying neural network.

our conditions, for a continuous $f : \mathcal{R}^n \to \mathcal{R}$, we have for $\overline{f}$ obtained by EP

$$^\alpha\overline{f}(x_1, \dots, x_n) = f(^\alpha x_1, \dots, {}^\alpha x_n)$$

for $x_i \in \mathcal{F}_{CI}(\mathcal{R})$, i.e. the operations on fuzzy sets can be performed by interval operations, see e.g. [12]. Furthermore, if f is monotone the interval operations are easy to compute.

We now describe the feedforward phase. The network dynamics is the classical one of the feedforward neural network. After a tupple of signals is presented the network transfers signals layer by layer. For $x_i \in \mathcal{F}_{CI}(\mathcal{R})$ we denote $^\alpha x_i = [^\alpha x_{Li}, {}^\alpha x_{Ri}]$. Suppose we have the output interval signals $y_i^{(l-1)} \in \mathcal{F}_{CI}(\mathcal{R})$ of all the neurons of the $(l-1)$–th layer at disposal. As mentioned above we may represent $y_i^{(l-1)}$ by a collection of intervals $[^\alpha y_{Li}^{(l-1)}, {}^\alpha y_{Ri}^{(l-1)}]$. The total input to the j–th neuron in the l-th layer is then given by

$$[^\alpha z_{Lj}^{(l)}, {}^\alpha z_{Rj}^{(l)}] = \sum_{i=1}^{m_{l-1}} w_{ij}^{(l)} [^\alpha y_{Li}^{(l-1)}, {}^\alpha y_{Ri}^{(l-1)}] - \theta_j^{(l)}[1, 1]$$

which leads by the interval arithmetic to

$$^\alpha z_{Lj}^{(l)} = \sum_{i=1, w_{ij}^{(l)} \geq 0}^{m_{l-1}} w_{ij}^{(l)\alpha} y_{Li}^{(l-1)} + \sum_{i=1, w_{ij}^{(l)} < 0}^{m_{l-1}} w_{ij}^{(l)\alpha} y_{Ri}^{(l-1)} - \theta_j^{(l)} \tag{1}$$

and

$$^\alpha z_{Rj}^{(l)} = \sum_{i=1, w_{ij}^{(l)} \geq 0}^{m_{l-1}} w_{ij}^{(l)\alpha} y_{Ri}^{(l-1)} + \sum_{i=1, w_{ij}^{(l)} < 0}^{m_{l-1}} w_{ij}^{(l)\alpha} y_{Li}^{(l-1)} - \theta_j^{(l)} \ . \tag{2}$$

As the transfer function $y(z) = \frac{1}{1+e^{-\lambda_j^{(l)} z}}$ is increasing for $\lambda_j^{(l)} > 0$ and decreasing for $\lambda_j^{(l)} < 0$ (note that for $\lambda_j^{(l)} = 0$, $y(z) = \frac{1}{2}$), we have

$$[^\alpha y_{Lj}^{(l)}, {}^\alpha y_{Rj}^{(l)}] = \left[\frac{1}{1 + e^{-\lambda_j^{(l)\alpha} z_{Lj}^{(l)}}}, \frac{1}{1 + e^{-\lambda_j^{(l)\alpha} z_{Rj}^{(l)}}} \right] \text{ for } \lambda_j^{(l)} \geq 0$$

and

$$[^\alpha y_{Lj}^{(l)}, {}^\alpha y_{Rj}^{(l)}] = \left[\frac{1}{1 + e^{-\lambda_j^{(l)\alpha} z_{Rj}^{(l)}}}, \frac{1}{1 + e^{-\lambda_j^{(l)\alpha} z_{Lj}^{(l)}}} \right] \text{ for } \lambda_j^{(l)} < 0 \ .$$

The described signal transfer is performed successively for the layers $l = 1, \dots, r$.

Return now to our problem from the functional point of view. For a given (regular) neural network which represents a mapping $F : \mathcal{R}^{m_0} \mapsto (0,1)^{m_r}$ we have obtained a network with fuzzy signals which represents a mapping $\widehat{F} : \mathcal{F}_{CI}(\mathcal{R})^{m_0} \mapsto \mathcal{F}_{CI}((0,1))^{m_r}$. A natural question at this point is what is the relationship of $\widehat{F}$ to the function $\overline{F}$ obtained from F by the EP. We will answer this question in general. We face the task of extending the function F which is in a certain way composed of several simple functions. Extension by EP can be done in two ways—extend the whole composite function (in our case to $\overline{F}$) or extend the simple constituent functions and compose them (in our case to $\widehat{F}$). In the two following statements we suppose that the structure of truth values forms a complete lattice of which the usual interval $[0,1]$ is a special case [14]. We need the following lemma.

Lemma 1 *Let $\mathcal{L} = \langle L, \vee, \wedge \rangle$ be a complete lattice, $K_i \subseteq L$, $i \in I$, $K \subseteq L$, such that $\bigcup\{K_i;\, i \in I\} = K$. Then it holds*

$$\bigvee\{\bigvee K_i;\, i \in I\} = \bigvee K \ .$$

Proof Denote $a = \bigvee K$, $a_i = \bigvee K_i$. We have to prove $\bigvee\{a_i;\, i \in I\} = a$. $K_i \subseteq K$ implies $a_i \leq a$ for all $i \in I$. Hence $\bigvee\{a_i;\, i \in I\} \leq a$. Conversely, for each $b \in K$ there is a j such that $b \in K_j$, and thus $b \leq a_j \leq \bigvee\{a_i;\, i \in I\}$. Hence $a = \bigvee K = \bigvee\{b \in K\} \leq \bigvee\{a_i;\, i \in I\}$. We have $\bigvee\{a_i;\, i \in I\} = a$. □

The following theorem shows the relationship of the two ways of extending a composite function.

Theorem 2 *Let $g : Y^m \to Z$, $f_i : X^k \to Y$, $i = 1, \dots, m$, be functions, $A_1, \dots, A_k$ fuzzy sets in X. Then for the corresponding extensions it holds*

$$\overline{g(f_1, \dots, f_m)}(A_1, \dots, A_k) \subseteq \overline{g}(\overline{f_1}, \dots, \overline{f_m})(A_1, \dots, A_k) \ .$$

If, moreover, $m = 1$, then even

$$\overline{g(f_1)}(A_1, \dots, A_k) = \overline{g}(\overline{f_1})(A_1, \dots, A_k)$$

holds.

Proof By the extension principle we have

$$\begin{aligned}
&\overline{g}(\overline{f_1}, \dots, \overline{f_m})(A_1, \dots, A_k)(z) = \overline{g}(\overline{f_1}(A_1, \dots, A_k), \dots, \overline{f_m}(A_1, \dots, A_k))(z) = \\
= \;&\overline{g}(\overline{f_1}(A_1, \dots, A_k) \times \cdots \times \overline{f_m}(A_1, \dots, A_k))(z) = \\
= \;&\overline{g}(\overline{f_1}(A_1 \times \cdots \times A_k) \times \cdots \times \overline{f_m}(A_1 \times \cdots \times A_k))(z) = \\
= \;&\bigvee\{(\overline{f_1}(A_1 \times \cdots \times A_k) \times \cdots \times \overline{f_m}(A_1 \times \cdots \times A_k))(y_1, \dots, y_m);\, g(y_1, \dots, y_m) = z\} = \\
= \;&\bigvee\{\bigwedge_{i=1}^{m}\{\bigvee\{A_1(x_1) \wedge \cdots \wedge A_k(x_k); f_i(x_1, \dots, x_k) = y_i\}\};\, g(y_1, \dots, y_m) = z\} = a
\end{aligned}$$

and

$$\begin{aligned}
&\overline{g(f_1, \dots, f_m)}(A_1, \dots, A_k)(z) = \overline{g(f_1, \dots, f_m)}(A_1 \times \cdots \times A_k)(z) = \\
= \;&\bigvee\{A_1(x_1) \wedge \cdots \wedge A_k(x_k);\, g(f_1, \dots, f_m)(x_1, \dots, x_k) = z\} = b
\end{aligned}$$

Denote

$$K = \{A_1(x_1) \wedge \cdots \wedge A_k(x_k);\ g(f_1, \ldots, f_m)(x_1, \ldots, x_k) = z\}\ ,$$

$$K_{yi} = \{A_1(x_1) \wedge \cdots \wedge A_k(x_k);\ \exists \vec{y} = \langle y_1, \ldots, y_i, \ldots, y_m\rangle \in g^{-1}(z),\ f_i(x_1, \ldots, x_k) = y_i = y\}\ .$$

We have $a = \bigvee_{z \in g^{-1}(y)} \bigwedge_{i=1}^{n} \bigvee_{yi} K_{yi}$ and $b = \bigvee K$. Clearly, $K \subseteq K_{yi}$ for each y and i. By well-known lattice properties it follows $\bigvee K \leq \bigvee_{yi} K_{yi}$, hence $\bigvee K \leq \bigwedge_{i=1}^{n} \bigvee_{yi} K_{yi}$, hence $b = \bigvee K \leq \bigvee_{\vec{y} \in g^1(z)} \bigwedge_{i=1}^{n} \bigvee_{yi} K_{yi} = a$ which proves the first part.
For $m = 1$ the assertion follows immediately from Lemma 1 putting $K_i = K_{y1}$. $\square$

From Theorem 2 we get by induction over the number of layers the following statement.

Corollary 3 *Let $x_1, \ldots, x_{m_0}$ be fuzzy sets in $\mathcal{R}$ and $\overline{F}(x_1, \ldots, x_{m_0}) = \langle \overline{y}_1, \ldots, \overline{y}_{m_r}\rangle$, $\widehat{F}(x_1, \ldots, x_{m_0}) = \langle \widehat{y}_1, \ldots, \widehat{y}_{m_r}\rangle$, where $\overline{F}$ and $\widehat{F}$ are the functions described above. Then for $i = 1, \ldots, m_r$ it holds*

$$\overline{y}_i \subseteq \widehat{y}_i\ .$$

It can be easily demonstrated [3] that the equality of the output fuzzy sets does not hold in general even in the case of our networks. In other words, the output fuzzy sets of $\widehat{F}$ are "wider" that the output fuzzy sets of $\overline{F}$. However, for crisp numbers taken as singletons the functions $\widehat{F}$, $\overline{F}$, and F coincide which has a positive consequence: If the inputs have one-element kernels (which is often the case) then the kernels of the outputs of $\widehat{F}$ and $\overline{F}$ coincide and are also one-element.

We further progress with the adaptation of our network. Given a training set $T = \{\langle X_p, O_p\rangle = \langle x_1, \ldots, x_n, o_1, \ldots, o_m\rangle \in \mathcal{F}_{CI}(\mathcal{R})^n \times \mathcal{F}_{CI}((0,1))^m, p \in P\}$ we want to design a network with $m_0 = n$, $m_r = m$, and adapt it to T. Adaptation in the case of feedforward networks means the design of a network which approximately represents the mapping partially prescribed by T. Hence we want $\widehat{F}$ to satisfy $\widehat{F}(X_p) \approx O_p$ for all $p \in P$ where $\approx$ is some further unspecified relation "approximately equals". We omit the phase of determining the number of layers and neurons in layers and suppose this has been done. Our task is to find appropriate parameters. We search not only for weights, but also for thresholds and slopes, see [13]. Adaptation leads to the minimization process of an appropriate error function for which we choose the function

$$E = \sum_{p \in P} \sum_{k=1}^{C} \beta(\alpha_k)\, {}^{\alpha_k}E^p$$

where

$${}^{\alpha}E^p = {}^{\alpha}E^p_L + {}^{\alpha}E^p_R = \tfrac{1}{2} \sum_{i=1}^{m_r} ({}^{\alpha}y^{(r)}_{Li} - {}^{\alpha}o^p_{Li})^2 + \tfrac{1}{2} \sum_{i=1}^{m_r} ({}^{\alpha}y^{(r)}_{Ri} - {}^{\alpha}o^p_{Ri})^2\ ,$$

and $\alpha_1 \leq \cdots \leq \alpha_C$, $\beta(\alpha_1) \leq \cdots \leq \beta(\alpha_C)$. The $\beta(\alpha_k)$ is the weight of importance of the α_k-cut, C represents the discretization. In order to minimize the error function one usually uses the method of the steepest descent approach. Recall that by the direction of the steepest descent it is meant the direction in which the function falls most rapidly by infinitesimal changes in this direction. We search in $\mathcal{R}^{W+2N}$, W is the number of weights, N is the number of neurons in the net, for the point $p = \langle \ldots, w^{(l)}_{ij}, \ldots, \theta^{(l)}_j, \ldots, \lambda^{(l)}_j, \ldots\rangle$ in which E takes the minimum. By the standard method of the steepest descent, one chooses an initial $p(0)$, e.g. at random, and then at any time t determines the change $\Delta p(t)$

$$\Delta p(t) = \eta d(p(t)) + \mu \Delta p(t-1)$$

where $d(p(t))$ is the direction of the steepest descent of E in the point $p(t)$, $\eta > 0$ is the length of the step (learning rate) and μ is a momentum constant.

We have naturally arrived at the point where interval networks, i.e. networks that work with real intervals instead of real numbers, see [2, 3, 18, 19], can be exploited. From (1) and (2) it is clear that the functions ${}^{\alpha}E^{p}$ (and hence E) are not differentiable in general. The author in [18] uses a little trick to get differentiable error function. Formulas (1) and (2) can be rewritten as follows,

$${}^{\alpha}z_{Lj}^{(l)} = \sum_{i=1}^{m_{l-1}} (sgn^{+}(w_{ij}^{(l)}) w_{ij}^{(l)} {}^{\alpha}y_{Li}^{(l-1)} + sgn^{-}(w_{ij}^{(l)}) w_{ij}^{(l)} {}^{\alpha}y_{Ri}^{(l-1)}) - \theta_{j}^{(l)}$$

and

$${}^{\alpha}z_{Lj}^{(l)} = \sum_{i=1}^{m_{l-1}} (sgn^{-}(w_{ij}^{(l)}) w_{ij}^{(l)} {}^{\alpha}y_{Li}^{(l-1)} + sgn^{+}(w_{ij}^{(l)}) w_{ij}^{(l)} {}^{\alpha}y_{Ri}^{(l-1)}) - \theta_{j}^{(l)}$$

where $sgn^{+}(x)$ is the signum function and $sgn^{-} = 1 - sgn^{+}$. The undifferentiable functions sgn^{+} and sgn^{-} are in [18] substituted by their differentiable analogies s^{+} and s^{-}, respectively, where

$$s^{+}(x) = \frac{1}{1+e^{-x}} \quad \text{and} \quad s^{-}(x) = \frac{1}{1+e^{x}} \ .$$

The function E is then differentiable, however, this approximation leads to an inaccurate interval arithmetic. The main disadvantage of this modified architecture which invalidates it for use for our purposes in networks with fuzzy signals is the subsethood nonmonotonicity, see [2]. We say that function mapping n-tupples of intervals (or fuzzy sets) to m-tupples of intervals (or fuzzy sets) is subsethood monotonic if for every interval inputs x_i, x_i', $i = 1, \ldots, n$ with $x_i \subseteq x_i'$ we have $y_j \subseteq y_j'$, $j = 1, \ldots, m$, for the corresponding outputs. The following proposition can be easily proved.

Proposition 4 *Let $\overline{f}$ be a function obtained from $f : \mathcal{R}^n \to \mathcal{R}$ by the Extension Principle. Then for every fuzzy sets x_i, x_i' in $\mathcal{R}$, $i = 1, \ldots n$, such that $x_i \subseteq x_i'$ we have $\overline{f}(x_1, \ldots x_n) \subseteq \overline{f}(x_1', \ldots x_n')$.*

By induction it follows that the mapping $\widehat{F}$ is subsethood monotonic. As a special case we obtain the subsethood monotonicity for intervals. It follows that the feedforward network with fuzzy signals has to be subsethood monotonic, i.e. the network with approximate sgn^{+}, sgn^{-} is not satisfactory. The practical result of using a nonmonotonic network could be that from an α–cut representation of input fuzzy sets one gets a system of α–cuts which do not represent any output fuzzy sets at all (it could happen ${}^{\alpha}\widehat{F}(x) \not\subseteq {}^{\beta}\widehat{F}(x)$ for $\beta < \alpha$ and $m_0 = m_r = 1$).

3 Adaptation

We now show that a backpropagation like steepest descent algorithm is possible and well justified. We need the following concepts. By a *signature* we mean a tupple $\Sigma = \langle \Sigma_1, \ldots, \Sigma_n \rangle$ where $\Sigma_i \in \{+, -, \pm\}$ for $i = 1, \ldots, n$. If Σ is a signature, then by R^{Σ} we denote the set $R^{\Sigma} = \{\langle x_1, \ldots, x^n \rangle \in \mathcal{R}^n;\ x_i \geq 0 \text{ for } \Sigma_i = +,\ x_i \leq 0 \text{ for } \Sigma_i = -,\ x_i \in \mathcal{R} \text{ for } \Sigma_i = \pm,\ i = 1, \ldots, n\}$. Call a function $f : \mathcal{R}^n \mapsto \mathcal{R}$ *Σ-differentiable in* $a \in \mathcal{R}^n$ if there are $c_1, \ldots, c_n \in \mathcal{R}$ such that

$\lim_{u\to 0, u\in\mathcal{R}^\Sigma} \frac{f(x+a)-f(a)-(c_1u_1+\cdots+c_nu_n)}{|u|} = 0$. Let Σ be a signature, $a \in \mathcal{R}^n$. A Σ-*neighborhood of* a is a subset $U_\Sigma(a) \subseteq \mathcal{R}^n$ such that there is a (regular) neighborhood $U(a)$ with $U_\Sigma(a) = U(a)\cap(a+\mathcal{R}^\Sigma)$ where $a+\mathcal{R}^\Sigma = \{a+u;\ u\in\mathcal{R}^\Sigma\}$. We say that $f:\mathcal{R}^n \mapsto \mathcal{R}$ *has partial derivatives in* $U_\Sigma(a)$ if for each $b \in U_\Sigma(a)$ it holds: if $b_i \neq a_i$ for some i then $\frac{\partial f(b)}{\partial x_i}$ exists, and if $b_i = a_i$ for some i then $\frac{\partial^{\Sigma_i} f(b)}{\partial^{\Sigma_i} x_i}$ exists. Let $f(x_1,\ldots,x_n):\mathcal{R}^n \mapsto \mathcal{R}$ have the derivatives $f_u'^{+}(a)$ in $a\in\mathcal{R}^n$ for every $u \in U \subseteq \mathcal{R}^n$. The vector $d \in U$ such that

$$f'^{+}_{\frac{d}{|d|}}(a) = \min\{f'^{+}_{\frac{u}{|u|}};\ u \in U\}$$

is (if exists) called the *direction of the steepest descent in a with respect to U*. Here $f_u'^{+}(a) = \lim_{t\to 0^+} \frac{f(a+tu)-f(a)}{t}$. It holds that if f is Σ-differentiable in a then f has partial derivatives in $U_\Sigma(a)$.

The properties necessary for the backpropagation like adaptation algorithm to be well justified are the following ones. The proofs follow from [2], our form of the error function and the fact that a sum of Σ-differentiable function is again a Σ-differentiable function. Complete proofs can be found in [3].

Theorem 5 ([3]) *$E(\ldots, w_{ij}^{(l)}, \ldots, \theta_j^{(l)}, \ldots, \lambda_j^{(l)}, \ldots)$ is continuous.*

Theorem 6 ([3]) *For a given $p = \langle \ldots, w_{ij}^{(l)}, \ldots, \theta_j^{(l)}, \ldots, \lambda_j^{(l)}, \ldots\rangle$ let Σ be a signature determined as follows: $\Sigma_{ij}^{w(l)} = \pm$ for $w_{ij}^{(l)} \neq 0$, $\Sigma_{ij}^{w(l)} \in \{+,-\}$ for $w_{ij}^{(l)} = 0$, $\Sigma_j^{\lambda(l)} = \pm$ for $\lambda_j^{(l)} \neq 0$, $\Sigma_j^{\lambda(l)} \in \{+,-\}$ for $\lambda_j^{(l)} = 0$, $\Sigma_j^{\theta(l)} = \pm$. Then $E(\ldots, w_{ij}^{(l)}, \ldots, \theta_j^{(l)}, \ldots, \lambda_j^{(l)}, \ldots)$ is Σ-differentiable in p.*

Theorem 7 ([2]) *Let $I \subseteq \{1,\ldots,n\}$, $f(x_1,\ldots,x_n):\mathcal{R}^n \mapsto \mathcal{R}$ be Σ-differentiable in $a \in \mathcal{R}^n$ for each Σ with $\Sigma_i = \pm$ for $i \in I$, $\Sigma_i \in \{+,-\}$ for $i \notin I$. Let $d = \langle d_1,\ldots,d_n\rangle$ be a vector determined as follows:*

For $i \in I$, $d_i = -\frac{\partial f(a)}{\partial x_i}$.

For $i \notin I$, let

$$\begin{aligned}
&d_i = 0 \textit{ for } \tfrac{\partial^- f(a)}{\partial^- x_i} \le 0 \textit{ and } \tfrac{\partial^+ f(a)}{\partial^+ x_i} \ge 0,\\
&d_i = -\tfrac{\partial^- f(a)}{\partial^- x_i} \textit{ for } \tfrac{\partial^- f(a)}{\partial^- x_i} > 0 \textit{ and } \tfrac{\partial^+ f(a)}{\partial^+ x_i} \ge 0,\\
&d_i = -\tfrac{\partial^+ f(a)}{\partial^+ x_i} \textit{ for } \tfrac{\partial^- f(a)}{\partial^- x_i} \le 0 \textit{ and } \tfrac{\partial^+ f(a)}{\partial^+ x_i} > 0,\\
&d_i = -\tfrac{\partial^- f(a)}{\partial^- x_i} \textit{ for } \tfrac{\partial^- f(a)}{\partial^- x_i} > 0 \textit{ and } \tfrac{\partial^+ f(a)}{\partial^+ x_i} < 0 \textit{ and } |\tfrac{\partial^- f(a)}{\partial^- x_i}| > |\tfrac{\partial^+ f(a)}{\partial^+ x_i}|,\\
&d_i = -\tfrac{\partial^+ f(a)}{\partial^+ x_i} \textit{ for } \tfrac{\partial^- f(a)}{\partial^- x_i} > 0 \textit{ and } \tfrac{\partial^+ f(a)}{\partial^+ x_i} < 0 \textit{ and } |\tfrac{\partial^- f(a)}{\partial^- x_i}| < |\tfrac{\partial^+ f(a)}{\partial^+ x_i}|.
\end{aligned}$$

Then d is the direction of the steepest descent in a with respect to $\mathcal{R}^n$.

By previous statements, a deepest descent adaptation algorithm in our case is as meaningful as the classical backpropagation. It follows from the previous considerations and theorems that the only thing remaining to be able to use the steepest descent approach is to get the partial derivatives of E. Since E is a sum of the functions ${}^\alpha E^p$ it is enough to derive the formulas for derivatives of ${}^\alpha E^p$: The derivative of E equals then the sum of the derivatives of ${}^\alpha E^p$. In what follows we omit for simplicity the indicies denoting the appropriate α–cuts and patterns, i.e. we write E instead of ${}^\alpha E^p$, similarly for E_L, E_R, $y_{Lj}^{(l)}, y_{Rj}^{(l)}$, $z_{Lj}^{(l)}, z_j^{(l)}$ etc.

First, we derive formulas for computing $\frac{\partial E}{\partial w_{ij}^{(l)}}$. Suppose $w_{ij}^{(l)} \neq 0$. Then we have

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \frac{\partial E_L}{\partial w_{ij}^{(l)}} + \frac{\partial E_R}{\partial w_{ij}^{(l)}},$$

$$\frac{\partial E_L}{\partial w_{ij}^{(l)}} = \frac{\partial E_L}{\partial y_{Lj}^{(l)}} \frac{\partial y_{Lj}^{(l)}}{\partial z_{\triangle j}^{(l)}} \frac{\partial z_{\triangle j}^{(l)}}{\partial w_{ij}^{(l)}} + \frac{\partial E_L}{\partial y_{Rj}^{(l)}} \frac{\partial y_{Rj}^{(l)}}{\partial z_{\nabla j}^{(l)}} \frac{\partial z_{\nabla j}^{(l)}}{\partial w_{ij}^{(l)}},$$

$$\triangle = \left\{ \begin{array}{l} L \text{ for } \lambda_j^{(l)} \geq 0 \\ R \text{ for } \lambda_j^{(l)} < 0 \end{array} \right. , \qquad \nabla = \left\{ \begin{array}{l} R \text{ for } \lambda_j^{(l)} \geq 0 \\ L \text{ for } \lambda_j^{(l)} < 0 \end{array} \right. , \tag{3}$$

$$\frac{\partial y_{\oplus j}^{(l)}}{\partial z_{\otimes j}^{(l)}} = \lambda_j^{(l)} y_{\oplus j}^{(l)} (1 - y_{\oplus j}^{(l)}) \qquad \text{for} \qquad \oplus, \otimes \in \{L, R\},$$

$$\frac{\partial z_{Lj}^{(l)}}{\partial w_{ij}^{(l)}} = \begin{array}{ll} y_{Li}^{(l-1)} & \text{for } w_{ij}^{(l)} > 0 \\ y_{Ri}^{(l-1)} & \text{for } w_{ij}^{(l)} < 0 \end{array} , \frac{\partial z_{Rj}^{(l)}}{\partial w_{ij}^{(l)}} = \begin{array}{ll} y_{Ri}^{(l-1)} & \text{for } w_{ij}^{(l)} > 0 \\ y_{Li}^{(l-1)} & \text{for } w_{ij}^{(l)} < 0 \end{array} .$$

Now, for the output layer $(l = r)$ we have

$$\frac{\partial E_L}{\partial y_{Lj}^{(l)}} = y_{Lj}^{(l)} - o_{Lj}, \qquad \frac{\partial E_L}{\partial y_{Rj}^{(l)}} = 0,$$

whereas for the inner layer $(l < r)$

$$\frac{\partial E_L}{\partial y_{Lj}^{(l)}} = \sum_{k=1}^{m_{(l+1)}} \left(\frac{\partial E_L}{\partial y_{Lk}^{(l+1)}} \frac{\partial y_{Lk}^{(l+1)}}{\partial z_{\triangle k}^{(l+1)}} \frac{\partial z_{\triangle k}^{(l+1)}}{\partial y_{Lj}^{(l)}} + \frac{\partial E_L}{\partial y_{Rk}^{(l+1)}} \frac{\partial y_{Rk}^{(l+1)}}{\partial z_{\nabla k}^{(l+1)}} \frac{\partial z_{\nabla k}^{(l+1)}}{\partial y_{Lj}^{(l)}} \right)$$

holds. Furthermore,

$$\frac{\partial z_{Lk}^{(l+1)}}{\partial y_{Lj}^{(l)}} = \left\{ \begin{array}{ll} w_{jk}^{(l+1)} & \text{for } w_{jk}^{(l+1)} > 0 \\ 0 & \text{else} \end{array} \right. , \frac{\partial z_{Rk}^{(l+1)}}{\partial y_{Lj}^{(l)}} = \left\{ \begin{array}{ll} w_{jk}^{(l+1)} & \text{for } w_{jk}^{(l+1)} < 0 \\ 0 & \text{else} \end{array} \right. .$$

Similarly,

$$\frac{\partial E_L}{\partial y_{Rj}^{(l)}} = \sum_{k=1}^{m_{(l+1)}} \left(\frac{\partial E_L}{\partial y_{Lk}^{(l+1)}} \frac{\partial y_{Lk}^{(l+1)}}{\partial z_{\triangle k}^{(l+1)}} \frac{\partial z_{\triangle k}^{(l+1)}}{\partial y_{Rj}^{(l)}} + \frac{\partial E_L}{\partial y_{Rk}^{(l+1)}} \frac{\partial y_{Rk}^{(l+1)}}{\partial z_{\nabla k}^{(l+1)}} \frac{\partial z_{\nabla k}^{(l+1)}}{\partial y_{Rj}^{(l)}} \right)$$

$$\frac{\partial z_{Lk}^{(l+1)}}{\partial y_{Rj}^{(l)}} = \left\{ \begin{array}{ll} w_{jk}^{(l+1)} & \text{for } w_{jk}^{(l+1)} < 0 \\ 0 & \text{else} \end{array} \right. , \frac{\partial z_{Rk}^{(l+1)}}{\partial y_{Rj}^{(l)}} = \left\{ \begin{array}{ll} w_{jk}^{(l+1)} & \text{for } w_{jk}^{(l+1)} > 0 \\ 0 & \text{else} \end{array} \right. .$$

To sum up, denote

$$\delta_{Lj}^{L,(l)} = \frac{\partial E_L}{\partial y_{Lj}^{(l)}} \lambda_j^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)}), \quad \delta_{Rj}^{L,(l)} = \frac{\partial E_L}{\partial y_{Rj}^{(l)}} \lambda_j^{(l)} y_{Rj}^{(l)} (1 - y_{Rj}^{(l)}).$$

Then we have

$$\frac{\partial E_L}{\partial w_{ij}^{(l)}} = \delta_{Lj}^{L,(l)} y_{\Box i}^{(l-1)} + \delta_{Rj}^{L,(l)} y_{\Diamond i}^{(l-1)}, \tag{4}$$

$$\Box = \begin{matrix} L \text{ for } w_{ij}^{(l)} > 0, \lambda_j^{(l)} \geq 0 \text{ or } w_{ij}^{(l)} < 0, \lambda_j^{(l)} \leq 0 \\ R \text{ for } w_{ij}^{(l)} > 0, \lambda_j^{(l)} < 0 \text{ or } w_{ij}^{(l)} < 0, \lambda_j^{(l)} > 0 \end{matrix} \tag{5}$$

$$\Diamond = \begin{matrix} R \text{ if } \Box = L \\ L \text{ if } \Box = R \end{matrix} \tag{6}$$

where for the output layer $(l = r)$

$$\delta_{Lj}^{L,(l)} = (y_{Lj}^{(l)} - o_{Lj}) \lambda_j^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)}) \ , \ \delta_{Rj}^{L,(l)} = 0$$

and for the inner layer $(l < r)$

$$\delta_{Lj}^{L,(l)} = \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{L,(l+1)} \zeta_{jk}^{(l+1)} + \delta_{Rk}^{L,(l+1)} \iota_{jk}^{(l+1)}) \lambda_j^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)})$$

$$\delta_{Rj}^{L,(l)} = \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{L,(l+1)} \iota_{jk}^{(l+1)} + \delta_{Rk}^{L,(l+1)} \zeta_{jk}^{(l+1)}) \lambda_j^{(l)} y_{Rk}^{(l)} (1 - y_{Rk}^{(l)})$$

where

$$\zeta_{jk}^{(l+1)} = \begin{cases} w_{jk}^{(l+1)} & \text{for } w_{jk}^{(l+1)} \lambda_k^{(l+1)} > 0 \\ 0 & \text{else} \end{cases} ,$$
$$\iota_{jk}^{(l+1)} = \begin{cases} w_{jk}^{(l+1)} & \text{for } w_{jk}^{(l+1)} \lambda_k^{(l+1)} < 0 \\ 0 & \text{else .} \end{cases}$$

Similar formulas can also be derived for $\frac{\partial E_R}{\partial w_{ij}^{(l)}}$.

$$\frac{\partial E_R}{\partial w_{ij}^{(l)}} = \delta_{Lj}^{R,(l)} y_{\Box i}^{(l-1)} + \delta_{Rj}^{R,(l)} y_{\Diamond i}^{(l-1)} \tag{7}$$

For the output layer $(l = r)$ we have

$$\delta_{Lj}^{R,(l)} = 0 \ , \ \delta_{Rj}^{R,(l)} = (y_{Rj}^{(l)} - o_{Rj}) \lambda_j^{(l)} y_{Rj}^{(l)} (1 - y_{Rj}^{(l)})$$

and for the inner layer ($l < r$),

$$\delta_{Lj}^{R,(l)} = \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{R,(l+1)} \zeta_{jk}^{(l+1)} + \delta_{Rk}^{R,(l+1)} \iota_{jk}^{(l+1)}) \lambda_j^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)})$$

$$\delta_{Rj}^{R,(l)} = \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{R,(l+1)} \iota_{jk}^{(l+1)} + \delta_{Rk}^{R,(l+1)} \zeta_{jk}^{(l+1)}) \lambda_j^{(l)} y_{Rk}^{(l)} (1 - y_{Rk}^{(l)})$$

If $w_{ij}^{(l)} = 0$ then for $\frac{\partial^+ E_L}{\partial w_{ij}^{(l)}}$ ($\frac{\partial^- E_L}{\partial w_{ij}^{(l)}}$, $\frac{\partial^+ E_R}{\partial w_{ij}^{(l)}}$ $\frac{\partial^- E_R}{\partial w_{ij}^{(l)}}$), the same formulas can be used except for (4), (7), where we set $\square, \diamond$ as in (5) for $w_{ij}^{(l)} > 0$ ($w_{ij}^{(l)} < 0$, $w_{ij}^{(l)} > 0$, $w_{ij}^{(l)} < 0$).

By an analogous way we derive formulas for $\frac{\partial E}{\partial \theta_j^{(l)}}$, $\frac{\partial E}{\partial \lambda_j^{(l)}}$.

$$\frac{\partial E}{\partial \theta_j^{(l)}} = \frac{\partial E_L}{\partial \theta_j^{(l)}} + \frac{\partial E_R}{\partial \theta_j^{(l)}}.$$

$$\frac{\partial E_L}{\partial \theta_j^{(l)}} = -\delta_{Lj}^{L,(l)} - \delta_{Rj}^{L,(l)}, \qquad \frac{\partial E_R}{\partial \theta_j^{(l)}} = -\delta_{Lj}^{R,(l)} - \delta_{Rj}^{R,(l)}$$

For $\lambda_j^{(l)} \neq 0$ we have

$$\frac{\partial E}{\partial \lambda_j^{(l)}} = \frac{\partial E_L}{\partial \lambda_j^{(l)}} + \frac{\partial E_R}{\partial \lambda_j^{(l)}}.$$

For the output layer ($l = r$) we have

$$\frac{\partial E_L}{\partial \lambda_j^{(l)}} = (y_{Lj}^{(l)} - o_{Lj}) z_{\triangle j}^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)})$$

$$\frac{\partial E_R}{\partial \lambda_j^{(l)}} = (y_{Rj}^{(l)} - o_{Rj}) z_{\nabla j}^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)})$$

For the inner layer ($l < r$),

$$\frac{\partial E_L}{\partial \lambda_j^{(l)}} = \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{L,(l+1)} \zeta_{jk}^{(l+1)} + \delta_{Rk}^{L,(l+1)} \iota_{jk}^{(l+1)}) z_{\triangle j}^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)}) + \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{L,(l+1)} \iota_{jk}^{(l+1)} + \delta_{Rk}^{L,(l+1)} \zeta_{jk}^{(l+1)}) z_{\nabla j}^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)})$$

$$\frac{\partial E_R}{\partial \lambda_j^{(l)}} = \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{R,(l+1)} \zeta_{jk}^{(l+1)} + \delta_{Rk}^{R,(l+1)} \iota_{jk}^{(l+1)}) z_{\triangle j}^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)}) + \sum_{k=1}^{m_{l+1}} (\delta_{Lk}^{R,(l+1)} \iota_{jk}^{(l+1)} + \delta_{Rk}^{R,(l+1)} \zeta_{jk}^{(l+1)}) z_{\nabla j}^{(l)} y_{Lj}^{(l)} (1 - y_{Lj}^{(l)})$$

Similarly as in the case of weights, if $\lambda_j^{(l)} = 0$ then for $\frac{\partial^+ E_L}{\partial \lambda_j^{(l)}}$, $\frac{\partial^- E_L}{\partial \lambda_j^{(l)}}$, $\frac{\partial^+ E_R}{\partial \lambda_j^{(l)}}$, $\frac{\partial^- E_R}{\partial \lambda_j^{(l)}}$, the same formulas as above can be used with the exception that for $\frac{\partial^+ E_L}{\partial \lambda_j^{(l)}}$ and $\frac{\partial^+ E_R}{\partial \lambda_j^{(l)}}$ we set $\triangle, \nabla$ as in (3) for $\lambda_j^{(l)} > 0$ whereas for $\frac{\partial^- E_L}{\partial \lambda_j^{(l)}}$, $\frac{\partial^- E_R}{\partial \lambda_j^{(l)}}$ we set $\triangle, \nabla$ as in (3) for $\lambda_j^{(l)} < 0$.

Note that the derived formulas can be a little bit simplified which is omitted in our paper because of its limited scope.

4 Possibilities of applications

The presented fuzzy neural network is an adaptive model mapping tupples of fuzzy sets to tupples of fuzzy sets. It can be applied in every situation where an input–output behaviour of a system which is partially described by input–output pairs of fuzzy sets is to be approximated. In the following we concentrate on the systems described by the so called linguistic IF–THEN rules.

Complex systems which are hardly to describe by traditional techniques can often be effectively described by means of fuzzy mathematics. Especially if there is only a linguistic description of the concerned problem expressed in the form of IF–THEN rules, methods of fuzzy logic have been proved to be very powerful. The expert knowledge of a system behaviour can often be represented by a set of IF–THEN rules of the form

$$\mathsf{IF}\ X\ \mathsf{is}\ \mathcal{A}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}\ .$$

Both the antecedent as well as the succedent can be of a more complex form. The linguistic expressions $\mathcal{A}$ and $\mathcal{B}$ (so called evaluating expressions) can be modeled by appropriate fuzzy sets A, B from $\mathcal{F}_{CI}(\mathcal{R})$. The whole knowledge about such a rule base is then to be aggregated into a suitable model R. For an input represented by the fuzzy set A' an output $B' = A' \circ R$ is then to be obtained by some method $\circ$. Nowadays, there are several methods developed in the frame of the so called approximated reasoning [9]. Usually, R is a fuzzy relation and $\circ$ is some projection operation. For a detailed discussion and formal analysis see e.g. [15]. Our model offers an alternative method: Design a network and adapt it to the training set T consisting of the associations $\langle A, B \rangle$. Then for a given input represented by A' we get an output B' from the network.

The great advantage of fuzzy logic systems is their interpretability, just the point where neural networks fail. On the other hand, the advantageous property of neural networks is their learning capability and generalization capability. It is very often the case that the linguistic description does not describe the problem completely. Rather, there are both the partial linguistic description and the measured data set describing the system behavior at disposal. In such a case we can transform the knowledge (linguistically expressed and crisp) to the training set T, design a network and adapt it to T. Our model is able to be adapted to both fuzzy and crisp data simultaneously. The net can be then used to perform the approximate reasoning task or, due to the evidenced good generalization property, it should be able to generate a new rule base or to complete the old one.

Concerning further the approximation reasoning task, of considerable importance is the computational complexity of the procedure performing the operation $\circ$, see [10, 6] for the analysis of some cases. In general, the complexity of $\circ$ increases with the number of rules in the rule base in the todays methods of approximate reasoning. On the other hand, the computational complexity of $\circ$ in the case of our network does not depend on the number of rules. It depends only on the architecture of the network, i.e. on the number of layers and neurons in the respective layers. We afford to omit the particular formulas for the computational complexity which are easy to derive.

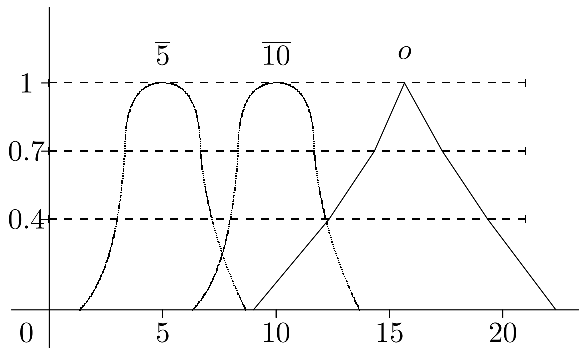

Figure 2: Inputs and corresponding output generated by the adapted network.

As a further advantageous property of our model can be considered the fact that due to the cut–by–cut processing of fuzzy sets it makes possible to get only a certain horizontal part of the output fuzzy set, e.g. the top part which is of considerable information value. This is not possible by the above discussed traditional methods.

5 Example

In this section we present an example demonstrating interpolation capabilities of our model. We trained a 2–layer network with two inputs, three neurons in the first layer, and one neuron in the second layer. The training set contained the following patterns: $T = \{\langle\overline{0},\overline{0},\overline{0}\rangle\langle\overline{0},\overline{20},\overline{20}\rangle\langle\overline{10},\overline{0},\overline{10}\rangle\langle\overline{10},\overline{20},\overline{30}\rangle\}$, where $\overline{r}$ denotes a fuzzy number "about r". After the network was adapted, inputs $\overline{5}$ and $\overline{10}$ have been presented to the network. The output o of the network is depicted in Fig. 2. (A linear transform has been used not to be restricted to the output interval $(0, 1)$.)

6 Conclusions

We have presented an adaptive feedforward neural network model which maps tupples of fuzzy sets to tupples of fuzzy sets. Our network is fuzzified by the extension principle in a way proposed in [4]. However, weights, thresholds and parameters of neuron transfer function are crisp. As shown, this leads to computational simple model with theoretically well justified and relatively simple adaptation formulas based on the steepest descent approach. The steepest descent adaptation eliminates the need for heuristic adaptation which was proposed e.g. in [8].

We have proposed applications in the approximate reasoning tasks and discussed the advantages and disadvantages of our model. From a point of view of further theoretical investigation a very interesting property establishing both neural networks and fuzzy systems as universal tools for various kinds of engineering applications is the universal approximation property. The authors in [5] negatively answered the question whether neural networks with both fuzzy signals and weights can be universal approximators. The question of approximation capabilities of the presented networks with fuzzy signals is still not definitely answered and is worth of further research. Both theoretical and application oriented investigations should show further possibilities in the systematic combination of fuzzy systems and neural networks. The aim of this paper was to contribute to this endeavour.

References

[1] Arbib M (Ed.) (1995) *The Handbook of Brain Theory and Neural Networks.* Cambridge, Massachusetts, London, MIT Press

[2] Bělohlávek R (1997) Backpropagation for interval patterns. Neural Network World. Int. J. on Neural and Mass–Parallel Computing and Information Systems. 7-3: 335–346

[3] Bělohlávek R (1998) *Networks Processing Indeterminacy.* PhD dissertation, Ostrava (available at request)

[4] Buckley J J, Hayashi Y (1992) Fuzzy Neural Nets and Applications. Fuzzy Sets and Artificial Intelligence. 1: 11–41

[5] Buckley J J, Hayashi Y (1994) Can fuzzy neural nets approximate continuous fuzzy functions? Fuzzy Sets & Systems. 61: 43–51

[6] Dvořák A (1997) Computational Properties of Fuzzy Logic Deduction. In: Reusch, B. (Ed.): Computational Intelligence. Theory and Applications. Proceedings of the 5th Fuzzy Days Dortmund, pp. 189–195, Berlin, Springer

[7] Gupta M M, Rao D H (1994) On the principles of fuzzy neural networks. Fuzzy Sets & Systems. 61: 1–18

[8] Hayashi Y, Buckley J J, Czogala E (1992) Direct Fuzzification of Neural Network and Fuzzified Delta Rule. Proc. of the 2nd Int. Conf. on Fuzzy Logic & Neural Networks, pp. 73–76, Iizuka, Japan

[9] Klir G J, Yuan B (1995) *Fuzzy Sets and Fuzzy Logic. Theory and Applications.* Upper Saddle River, NJ, Prentice Hall

[10] Kóczy L T (1995) Algorithmic aspects of fuzzy control. Int. J. of Approximate Reasoning. 12: 159–219

[11] Kosko B (1991) *Neural Networks and Fuzzy Systems.* Upper Saddle River, NJ, Prentice Hall

[12] Kruse R, Gebhardt J, Klawonn F (1995) *Fuzzy–Systeme.* Stuttgart, B. G. Teubner

[13] Kufudaki O, Hořejš J (1991) PAB: Parameters adapting back propagation. Neural Network World. Int. J. on Neural and Mass–Parallel Computing and Information Systems. 1: 267–274

[14] Novák V (1989) *Fuzzy Sets and Their Applications.* Bristol, Adam–Hilger

[15] Novák V (1996) Paradigm, formal properties and limits of fuzzy logic. Int. J. of General Systems. 24: 377–405

[16] Pedrycz W (1993) Fuzzy neural networks and neurocomputations. Fuzzy Sets & Systems. 56: 1–28

[17] Rumelhart D E, Hinton G E, Williams R J (1986) Learning representations by back-propagating errors. Nature. 323: 533–536

[18] Šíma J (1992) Generalized back propagation for interval training patterns. Neural Network World. Int. J. on Neural and Mass–Parallel Computing and Information Systems. 2: 167–173

[19] Šíma J (1995) Neural expert systems. Neural Networks. 8: 261–271