VERSITA

Int. J. Appl. Math. Comput. Sci., 2012, Vol. 22, No. 2, 449–459
DOI: 10.2478/v10006-012-0034-5

amcs

BACKPROPAGATION GENERALIZED DELTA RULE FOR THE SELECTIVE ATTENTION SIGMA–IF ARTIFICIAL NEURAL NETWORK

MACIEJ HUK

Institute of Informatics
Wrocław University of Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: maciej.huk@pwr.wroc.pl

In this paper the Sigma-if artificial neural network model is considered, which is a generalization of an MLP network with sigmoidal neurons. It was found to be a potentially universal tool for automatic creation of distributed classification and selective attention systems. To overcome the high nonlinearity of the aggregation function of Sigma-if neurons, the training process of the Sigma-if network combines an error backpropagation algorithm with the self-consistency paradigm widely used in physics. But for the same reason, the classical backpropagation delta rule for the MLP network cannot be used. The general equation for the backpropagation generalized delta rule for the Sigma-if neural network is derived and a selection of experimental results that confirm its usefulness are presented.

Keywords: artificial neural networks, selective attention, self consistency, error backpropagation, delta rule.

1. Introduction

In nature, selective attention is a mechanism which provides living organisms with the possibility to sift incoming data to extract information which is most important at a given moment and which should be processed in detail (Broadbent, 1982; Treisman, 1960). When limited processing capabilities do not allow rapid analysis of the whole scene of visual and other senses, selective attention can be viewed as a strategy of dynamical input space selection for gaining predefined goals by the system (e.g., an organism) interacting with a very complicated environment (Noton and Stark, 1971; Tsotsos *et al*., 2001; Vanrullen and Koch, 2003). Accordingly, selective attention systems are found to be very interesting from a theoretical point of view, and also as tools for many practical applications, such as analysis of large data sets, real time route planning for autonomic robots in dynamical environments, and dispersed sensor networks control (Desimone and Duncan, 1995; Olshausen *et al*., 1993; Houghton and Tipper, 1996; Hager and Toyama, 1999; Stark *et al*., 2000; Körding and König, 2001; Gupta, 2008; Indiveri, 2008; Ferguene and Toumi, 2009; Pedro and Dahunsi, 2011).

As most of the selective attention systems observed in nature use neuronal control mechanisms, many researchers try to realize selective attention solutions by using artificial neural networks. Unfortunately, networks that use higher-order neuron models, such as Sigma-Pi (Feldman and Ballard, 1982; Rumelhart *et al*., 1986; Mel, 1990; Olshausen *et al*., 1993), Power Unit (Durbin and Rumelhart, 1990) or Clusteron (Mel, 1992), realize only a very limited set of attentional mechanisms (Neville and Eldridge, 2002; Weber and Wermter, 2007).

Thus it can be very interesting that selective attention functionality, which seems to effectively mimic low-level attentional processes observed in humans, was found in a recently developed simple generalization of the well-known MLP network called *Sigma-if* (Huk, 2004; 2006; 2009). However, the Sigma-if neural network model to be trainable with use of the backpropagation algorithm (typical for MLP) needs a new, generalized form of the delta rule that will take care of the noncontinuous character of the aggregation functions of the Sigma-if neurons.

2. Preliminaries

The Sigma-if neural network is a type of synchronous, feedforward multilayer Artificial Neural Network (ANN) and possesses selective attention abilities (Niebur *et al.*, 2002; Huk, 2004; 2006). Such a neural network does not use separate centralized attention guidance modules. Its ability to realize low-level selective attention functional-

ity emerges as an effect of synergy between its hidden, Sigma-if neurons. Each Sigma-if neuron is a special direct generalization of a sigmoidal neuron which implements basic selective attention functionality via input connections grouping and stepwise conditional input signal accumulation. This is due to the new neuron's aggregation function (Duch and Jankowski, 1999; Huk, 2004).

Formally speaking, N dendrites of the Sigma-if neuron are divided into K distinct groups, by complementing each i-th input connection with an additional integer parameter $\theta_i \in \{0, 1, \ldots, K-1\}$, determining membership in one of the groups. This allows us to divide the process of signal accumulation into *K* steps, where *K* is a function of the neuron's grouping vector $\overline{\theta}^T$=$[\theta_1, \theta_2, \ldots, \theta_N]$:

$$K(\overline{\theta}) = \max_{i=1}^{N}(\theta_i). \tag{1}$$

During each step k (from 0 to $K-1$), the neuron accumulates data belonging to one selected group, such that

$$\theta_i = k. \tag{2}$$

Within each k-th group, partial activation $\Delta\varphi(k)$ is determined as a weighted sum of input signals and the appropriate Kronecker delta:

$$\Delta\varphi_k(\overline{w}, \overline{x}, \overline{\theta}) = \sum_{i=1}^{N} w_i x_i \delta(k, \theta_i), \tag{3}$$

where w_i and x_i are coefficients of the neuron's weight vector $\overline{w}$ and an input vector $\overline{x}$. This process is repeated until the activation derived from respective groups exceeds a preselected aggregation threshold φ^*. It can be described by the following recursive formula (vectors $\overline{w}$, $\overline{x}$ and $\overline{\theta}$ are omitted for clarity):

$$\varphi_k = \begin{cases} \Delta\varphi_k H(\varphi^* - \varphi_{k-1}) + \varphi_{k-1} & \text{if} \quad k \geq 0, \\ 0 & \text{if} \quad k < 0\,, \end{cases} \tag{4}$$

where H is Heaviside's function. This sum is then treated as the neuronal activation value. The input from remaining (heretofore unconsidered) groups is neglected. Thus, the form of the aggregation function $\varphi_{\text{Sigma-if}}$ is

$$\varphi_{\text{Sigma-if}}(\overline{w}, \overline{x}, \overline{\theta}) = \varphi_K(\overline{w}, \overline{x}, \overline{\theta}). \tag{5}$$

In the final stages of determining the output value Y of the neuron, the function (5) serves as a parameter of the nonlinear threshold (e.g., sigmoidal) function F:

$$Y(\overline{w}, \overline{x}, \overline{\theta}) = F(\varphi_{\text{Sigma-if}}(\overline{w}, \overline{x}, \overline{\theta})). \tag{6}$$

It is worth noting that the described model assumes that the state graph used during signal aggregation is always a simple directed path of nonterminal nodes corresponding to the accumulation procedure of neural activation. In a general case, the Sigma-if neuron, besides the vector of weights $\overline{w}$, includes one real parameter for the aggregation threshold φ^*, and an additional vector $\bar{\theta}$ collecting connections with only one nominal coefficient for each neuronal input connection.

In comparison with MLP neural network training, searching for a globally optimal set of the Sigma-if network parameters would be very computationally challenging. This is due to the noncontinuous character of the Sigma-if neuron grouping vector. While there is no quick and effective method of global searching for network weights and grouping vectors, one can assume that, at each Sigma-if neuron, coefficients of the grouping vector $\overline{\theta}$ are in fact direct functions of the weight vector. In the work of Huk (2004) the proposed solution is to sort inputs of a given Sigma-if neuron by their weights, and assign $\lceil N/K \rceil$ connections with the highest weights to the most significant group θ_1, next $\lceil N/K \rceil$ connections to group θ_2, and so on.

In the above solution, the mutual relationship between connection weights and grouping vectors allows an improvement of the backpropagation algorithm by the application of the self-consistency idea widely used in physics (Noh *et al.*, 1991; Fonseca *et al.*, 1998; Raczkowski *et al.*, 2001). To realize that, Sigma-if network training begins with random values of connection weights and with all connections assigned to the single group. This assures that at the beginnig of the training process the network behaves as a multilayer ANN with sigmoidal neurons, and all of the connections between neurons are treated as equally important. Then Sigma-if network connection weights are changed by an error backpropagation algorithm for ω training epochs without changes in grouping vectors. After ω training epochs, actual grouping vectors are computed for all Sigma-if neurons and then connection weights are changed again by the error backpropagation algorithm for the next ω training epochs. This process is repeated until the resulting network meets the stop condition of the backpropagation method.

Such alternate changes of two mutually dependant sets of parameters of the Sigma-if model can lead to optimization of both weights and grouping vectors even if changes of only one of these sets (e.g., weights) are directly guided by a known optimization algorithm. Thus the only element needed to implement such a process for a Sigma-if neural network is to know the generalized delta rule for this model.

3. Backpropagation delta rule for the multilayer feedforward neural network

It is convenient to show the derivation of a generalized delta rule for Sigma-if neural network in comparison with a backpropagation generalized delta rule for the MLP network. Thus, regardless of common knowledge about the

backpropagation algorithm, first we need to recall elements of this method (Rumelhart *et al*., 1986; Korbicz *et al*., 1994). This will simplify further parts of the derivation and will serve as a definition of a common set of symbols.

Let us consider the general case of a multilayer feedforward neural network with a full network of connections between neurons in adjacent layers, and a nondecreasing and differentiable activation function in individual neurons. To establish the symbols, we assume that every μ-th learning pattern is a pair containing the input vector $x^{z\mu}$ and the corresponding output vector $y^{z\mu}$. Simultaneously, consecutive layers of the network are numbered with index m and values from 1 to M, where layer m consist of n_m neurons. Consequently, the weight of the connection between the j-th neuron in the m-th layer and the i-th neuron of the previous layer is written as w_{ji}^m (in the case of double lower indices, the left subscript is the number of the neuron in the layer of the number indicated in the superscript, while the right subscript is the index of the neuron input). Similarly, the values of the aggregation function φ and the activation function $F(\varphi)$ for the j-th neuron of the m-th layer are denoted, respectively, by $\varphi_j^{m\mu}$ and $u_j^{m\mu}$, while for the i-th neuron of the input layer, which by definition realizes an identity transfer function, $u_i^{1\mu}$ is equal to $x_i^{z\mu}$.

Using the above notation and assuming that all neurons of network hidden layers are sigmoidal (with the aggregation function being a linear combination of input values and weights connections), we get the output values of neurons in the form

$$u_j^{m\mu} = F(\varphi_j^{m\mu}) = F\left(\sum_{i=1}^{n_m} w_{j,i}^m u_i^{(m-1)\mu}\right). \tag{7}$$

Operation of the backpropagation algorithm comes down to a cyclic repetition of four main phases. Using the designations made above, we can write that in each t-th cycle of the training process the phases for each μ-th training vector are as follows:

1. Provide the μ-th training vector for the network inputs and determine the value $u_j^{m\mu}$ of the output of each j-th neuron, in all layers of the network—from inputs to outputs (for m = 1,2, ...,M).

2. Calculate the value of the error $\delta_j^{M\mu}$ for each of n_M output neurons and the sum ξ_μ of their squares.

3. Propagate the output error backward from outputs to inputs with calculation of errors $\delta_j^{m\mu}$ for all neurons in hidden layers (for $m = M, M-1, \ldots, 2$).

4. Modify connection weights, starting from the output layer and ending in the input layer, according to the generalized delta rule for sigmoidal neurons and with the formula

$$w_{ji}^{m(t+1)} = w_{ji}^{m(t)} + \Delta w_{ji}^m. \tag{8}$$

After presenting all the training vectors the stopping condition of the algorithm is checked and, if it is not met, all the above steps are repeated in the next training cycle.

Leaving aside the question of the maximum allowable number of algorithm cycles, a typical backpropagation stopping condition is to determine whether the neural network output error for all vectors is lower than a given threshold. The network output error for a given μ-th training vector is a sum of squares of output neuron error values, given by the formula

$$\xi_\mu = \frac{1}{2}\sum_{j=1}^{n_M}(y_j^{z\mu} - u_j^{M\mu})^2. \tag{9}$$

We can thus define an error created in the j-th neuron of the m-th layer as

$$\delta_j^{m\mu} = -\frac{\partial \xi_\mu}{\partial \varphi_j^{m\mu}}, \tag{10}$$

which can be converted to the form

$$\delta_j^{m\mu} = -\frac{\partial \xi_\mu}{\partial u_j^{m\mu}}\frac{\partial u_j^{m\mu}}{\partial \varphi_j^{m\mu}} = -\frac{\partial \xi_\mu}{\partial u_j^{m\mu}}F'(\varphi_j^{m\mu}). \tag{11}$$

For the output layer we can directly write

$$\frac{\partial \xi_\mu}{\partial u_j^{M\mu}} = -(y_j^{z\mu} - u_j^{M\mu}). \tag{12}$$

In the case of hidden layers, an analogous partial derivative is, however, a bit more troublesome to calculate, due to the complexity of the dependence of ξ_μ on $u_j^{m\mu}$. To perform necessary transformations, one should use the dependence of the neuron aggregation function $\varphi_l^{(m+1)\mu}$ in layer $m+1$ on the value of $u_j^{m\mu}$. But by taking into account all contributions of the corresponding changes in aggregation functions to the change in the network error, and by using the chain rule of differentiation of composite functions, we obtain

$$\frac{\partial \xi_\mu}{\partial u_j^{m\mu}} = \sum_{l=1}^{n_{m+1}} \frac{\partial \xi_\mu}{\partial \varphi^{(m+1)\mu}{}_l}\frac{\partial \varphi_l^{(m+1)\mu}}{\partial u_j^{m\mu}}. \tag{13}$$

Recalling now (10) and performing the differentiation of (7) with respect to $u_j^{m\mu}$, we can finally write

$$\frac{\partial \xi_\mu}{\partial u_j^{m\mu}} = -\sum_{l=1}^{n_{m+1}} \delta_j^{(m+1)\mu} w_{lj}^{(m+1)\mu}. \tag{14}$$

On the basis of Eqns. (11), (12) and (14), we can assign each neuron of a multilayer network a suitable output

error value. For the output layer, the error of the neuron output is given by

$$\delta_j^{M\mu} = F'(\varphi_j^{M\mu})(y_j^{z\mu} - u_j^{M\mu}), \tag{15}$$

and in the case of hidden neurons their output error has the form

$$\delta_j^{m\mu} = F'(\varphi_j^{m\mu}) \sum_{l=1}^{n_{m+1}} \delta_j^{(m+1)\mu} w_{lj}^{(m+1)\mu}. \tag{16}$$

However, to specify the relevant rule of changing connection weight w_{ji}^m in the direction of the error gradient in the space of weights, which would provide improved network operation in the next step of the training algorithm, we have to determine the value of the expression

$$\Delta w_{ji}^m = -\eta \frac{\partial \xi_\mu}{\partial w_{ji}^m} = -\eta \frac{\partial \xi_\mu}{\partial \varphi_j^{m\mu}} \frac{\partial \varphi_j^{m\mu}}{w_{ji}^m}. \tag{17}$$

Equation (7) shows that the second partial derivative occurring on the right-hand side of (17) is equal to $u_i^{(m-1)\mu}$. Moreover, its first partial derivative on the basis of (11) can be written as

$$\frac{\partial \xi_\mu}{\partial \varphi_j^{m\mu}} = \frac{\partial \xi_\mu}{\partial u_j^{m\mu}} \frac{\partial u_j^{m\mu}}{\partial \varphi_j^{m\mu}} = -\delta_j^{m\mu}. \tag{18}$$

Thus, we finally get a generalized form of the delta rule:

$$\Delta w_{ji}^m = \eta \delta_j^{m\mu} u_i^{(m-1)\mu}, \tag{19}$$

while for the output neurons it is expressed as

$$\Delta w_{ji}^M = \eta u_i^{(M-1)\mu} F'(\varphi_j^{M\mu})(y_j^{z\mu} - u_j^{M\mu}) \tag{20}$$

and for hidden neurons as

$$\Delta w_{ji}^m = \eta u_i^{(m-1)\mu} F'(\varphi_j^{m\mu}) \sum_{l=1}^{n_{m+1}} \delta_j^{(m+1)\mu} w_{lj}^{(m+1)\mu}. \tag{21}$$

As the effect of the use of the above set of expressions, in each cycle of the backpropagation algorithm the neural network parameters are changed in the direction of the largest possible decrease in the error function. As a result, repeated presentation of all training vectors (at each cycle, if possible, in different order) leads to local minimization of the error function, while the size of the optimization steps is steered by the parameter η, often called the learning factor.

4. Generalized delta rule for the Sigma-if neural network

For a multilayer Sigma-if neural network, the first two phases of the backpropagation algorithm—computation of the network output values and determination of neurons' output errors—almost do not change in comparison with a multilayer ANN with sigmoidal neurons. The method of calculating the error components $\delta_j^{M\mu}$ for the output layer and the μ-th training vector remains unchanged as a result of the independence of the derivative of (10) of the form of aggregation functions of network output neurons. As a result, the function (9), determining the mean square error over all outputs of the neural network, remains unmodified. In turn, the main difference in Sigma-if network training is the need to memorize for each j-th Sigma-if neuron in the m-th layer the number $k_j^{*m\mu}$ of groups of input connections activated during its output computation for the μ-th training vector. Thus, looking one more time at the definition (5) we can formally write that in an interesting case of low-level selective attention, when not all input connections are used to compute neuron output value,

$$\exists\, k_j^{*m\mu} < K : \varphi_{k_j^{*m\mu}}(\overline{w}, \overline{x}, \overline{\theta}) \geq \varphi^*. \tag{22}$$

The values k^* are also essential for proper execution of the error backpropagation procedure, as they keep information about which input connections of the given neuron influenced its output for the given training vector. These values also allow rewriting the definition of the Sigma-if aggregation function (5) in the non-recursive form:

$$\begin{aligned} \varphi_{\text{Sigma-if}}(\overline{w}, \overline{x}, \overline{\theta}) &= \sum_{k=1}^{k^*} \Delta\varphi_k(\overline{w}, \overline{x}, \overline{\theta}) \\ &= \sum_{k=1}^{k^*} \sum_{i=1}^{N} w_i x_i \delta(k, \theta_i), \end{aligned} \tag{23}$$

which is useful in practical implementations and, which is more important, it will be needed during further formal transformations.

Due to the use of the aggregation function $\varphi_{\text{Sigma-if}}$ during the error backpropagation phase, the method of determining the errors in the output neurons undergoes a formal change. It can be shown for the aggregation function given by the expression (23), by replacing the number of neuron inputs N and neuron input values x_i with the number n_m of neurons in the previous layer m and their output values $u_j^{m\mu}$, respectively, and by calculating again the derivative (13) (in the case of double lower indices, the left subscript is the number of a neuron in the layer of the number indicated in the superscript, while the right subscript is the number of the neuron input; for simplicity, the 'Sigma-if' subscript of the aggregation function is further

omitted):

$$\frac{\partial \varphi_l^{(m+1)\mu}}{\partial u_j^{m\mu}} = \frac{\partial}{\partial u_j^{m\mu}} \sum_{k=1}^{k_l^{*(m+1)\mu}} \sum_{i=1}^{n_m} \left(w_{l,i}^{m+1} u_i^{m\mu} \delta(k, \theta_{l,i}^{m+1}) \right). \tag{24}$$

Hence, after expanding the sum over k and performing the differentiation of the right-hand side, the above equation takes the form

$$\begin{aligned} &\frac{\partial}{\partial u_j^{m\mu}} \left(\sum_{i=1}^{n_m} w_{l,i}^{m+1} u_i^{m\mu} \delta(1, \theta_{l,i}^{m+1}) + \dots \right. \\ &\quad \left. + \dots + \sum_{i=1}^{n_m} w_{l,i}^{m+1} u_i^{m\mu} \delta(k_l^{*(m+1)\mu}, \theta_{l,i}^{m+1}) \right) \\ &= w_{l,j}^{m+1} \delta(1, \theta_{l,j}^{m+1}) + w_{l,j}^{m+1} \delta(2, \theta_{l,j}^{m+1}) + \dots \\ &\quad + \dots + w_{l,j}^{m+1} \delta(k_l^{*(m+1)\mu}, \theta_{l,j}^{m+1}). \end{aligned} \tag{25}$$

Then, by factoring out the common weight terms, we can write

$$\frac{\partial \varphi_l^{(m+1)\mu}}{\partial u_j^{m\mu}} = w_{l,j}^{m+1} \sum_{k=1}^{k_l^{*(m+1)\mu}} \delta(k, \theta_{l,j}^{m+1}). \tag{26}$$

However, the sum of Kronecker deltas appearing on the right-hand side of (26) may take only two values: one when the j-th input of the l-th neuron belongs to one of the groups active during signal aggregation for the vector μ, and zero otherwise. In the first case, the component of the $\theta_{l,j}^{m+1}$ grouping vector assigned to the j-th input connection is less than or equal to the number of active groups $k_l^{*(m+1)\mu}$, and in the second one it is greater than this value. This allows us to conclude that

$$\frac{\partial \varphi_l^{(m+1)\mu}}{\partial u_j^{m\mu}} = w_{l,j}^{m+1} H(k_l^{*(m+1)\mu} - \theta_{l,j}^{m+1}). \tag{27}$$

Finally, by applying the derivative calculated in this way to (13), one can determine the formula for the output error of j-th neuron in the m-th hidden layer of the Sigma-if neural network (based on (11)):

$$\delta_j^{m\mu} = F'(\varphi_j^{m\mu}) \cdot \sum_{l=1}^{n_{m+1}} \left(\delta_l^{(m+1)\mu} w_{l,j}^{m+1} H(k_l^{*(m+1)\mu} - \theta_{l,j}^{m+1}) \right), \tag{28}$$

where the parameter l enumerates consecutive neurons in layer $m+1$.

The above expression differs from the corresponding formula (16) for the multilayer feedforward network with sigmoidal neurons only by the appearance of the Heaviside function. Due to this change, when not all inputs of the Sigma-if neuron are involved in determining its output value, during the backpropagation phase the error is propagated only by the connections that were used. However, this is fully consistent with the idea of the backpropagation algorithm. Neuron connections inactive during the aggregation of the input signals, despite non-zero weights and availability of signals, do not make any contribution to the activation of a neuron, and consequently, they do not influence the Sigma-if neurons output error values. Thus the weights of inactive connections should not be changed.

To determine the general rule of weight modification in the network of Sigma-if neurons, one should calculate the expression (17) with the use of Eqn. (28). Therefore, the following derivative requires consideration:

$$\frac{\partial \varphi_j^{m\mu}}{\partial w_{j,i}^m} = \frac{\partial}{\partial w_{j,i}^m} \sum_{k=1}^{k_j^{*m\mu}} \sum_{i=1}^{n_m} w_{j,i}^m u_i^{(m-1)\mu} \delta(k, \theta_{j,i}^m). \tag{29}$$

However, it is easy to note the similarity between the above expression and the formula (24). By analogy, without unnecessary transformations, we get

$$\frac{\partial \varphi_j^{m\mu}}{\partial w_{j,i}^m} = u_i^{(m-1)\mu} H(k_j^{*m\mu} - \theta_{j,i}^m). \tag{30}$$

As a result, the generalized delta rule specifying the change in the weight value of the i-th input of the j-th neuron in the m-th Sigma-if network layer takes the form

$$\Delta w_{j,i}^m = \eta \delta_j^{m\mu} u_i^{(m-1)\mu} H(k_j^{*m\mu} - \theta_{j,i}^m), \tag{31}$$

where $u_i^{(m-1)\mu}$ is the output value of the i-th neuron in m-1 layer for training vector μ, and η is a learning factor.

Finally, after taking into account the relevant formulas for errors of different elements of the Sigma-if network, the generalized delta rule for the output layer of its neurons is given by

$$\Delta w_{j,i}^M = \eta u_i^{(M-1)\mu} H(k_j^{*M\mu} - \theta_{j,i}^M) F'(\varphi_j^{M\mu})(y_j^{z\mu} - y_j^\mu), \tag{32}$$

while its counterpart for the hidden layers of Sigma-if neurons is

$$\begin{aligned} \Delta w_{j,i}^m &= \eta u_i^{(m-1)\mu} H(k_j^{*m\mu} - \theta_{j,i}^m) F'(\varphi_j^{m\mu}) \\ &\quad \cdot \sum_{l=1}^{n_{m+1}} \left(\delta_l^{(m+1)\mu} w_{l,j}^{m+1} H(k_l^{*(m+1)\mu} - \theta_{l,j}^{m+1}) \right). \end{aligned} \tag{33}$$

The Heaviside function appearing in the expression (31) can be viewed as a mechanism that counteracts unnecessary modifications of the network structure in those

parts which are not used for determining the output values of individual neurons for a given training vector. Thus, both in the hidden and the output layer, the weights of connections that were inactive during the process of input signals accumulation are not modified.

5. Results of experiments

The generalized delta rule for the Sigma-if neuron and its conditional aggregation function presented above was additionally examined by verification of the whole Sigma-if network properties using example classification tasks of selected benchmark problems of the UCI Machine Learning Repository. During tests, simulated Sigma-if neural networks were compared with MLP networks with the same architectures (one hidden layer, the number of neurons in layers dependent on the solved problem—see figures below). Their generalization abilities were additionally analysed against the best results of other machine learning classification methods (see, e.g., Huk, 2006). As the sigmoidal neuron is a special case of a Sigma-if neuron, multilayer networks with sigmoidal neurons were simulated by Sigma-if networks with the number of inputs groups K of all Sigma-if neurons set to one. In all cases, standard input signal coding was used, output coding was bipolar and answers of the neural network were computed in the winner-takes-all manner.

Along with classification accuracies u for training and γ for test data, properties such as the neural network data processing time τ as well as hidden connections and network input activity (designated by hca and nia, respectively) were considered. Hidden connections activity hca and network inputs activity nia were used to represent the percentage ratio of the number of hidden and input connections used during data processing, compared with all of the network's hidden and input connections, respectively. These parameters allowed checking if hidden Sigma-if neurons use their selective attention ability in practice. For the completeness of the analysis, for each given problem and trained network, the percentage of all inputs used to classify all test vectors niu was calculated. This was important in order to determine if selective attention functionality is also realized at the level of the whole Sigma-if network. All measured values were calculated as averaged outcomes of ten independent 10-fold cross validations.

To precisely check how selective attention abilities of the Sigma-if network influence the properties of the resulting models, beside generalization γ of the networks that were final results of each training, classification performance was measured also for each network model generated in each step of backpropagation during validation steps. This allowed finding out how selective attention changes maximal classification accuracy of test data (γ_m) reachable by the networks generated during one average training. For networks with the greatest γ_m, also classification accuracy for training data (u_m) was measured. Again, to reduce the influence of initial network weights selection on the results, all classification accuracies were averaged for all steps of ten independent 10-fold cross validations. It must be stressed that classification performance of networks measured during backpropagation was not used to control the training process. The reason to collect additional data was checking for possible unpredicted influence of using Sigma-if neurons on the course of the training process.

During experiments, also the average data processing time τ of the input vector for all trained networks was measured to check relative data processing costs of MLP and Sigma-if networks. All time measurements were conducted on a single dedicated computer with a 2.6 GHz processor. Regardless of the very precise time measurement procedure used, actual timings on other hardware setups may vary considerably. But the presented results can still be used to show the order of possible gains for time-critical applications if the Sigma-if network is used.

Other parameters, such as the aggregation threshold φ^* and the grouping vector actualization interval ω, were set to 0.6 and 25, respectively, following preliminary tests. During those tests, also the number of hidden neurons for each problem was preselected to the value for which a multilayer feedforward neural network with sigmoidal neurons achieved the lowest average generalization error during ten independent 10-fold cross validations. The backpropagation stop condition, identical in all experiments, was using two constant thresholds to check if the training algorithm reached given the classification accuracy of training data or the maximal number of 8000 training epochs.

The obtained results indicate that increasing the num-

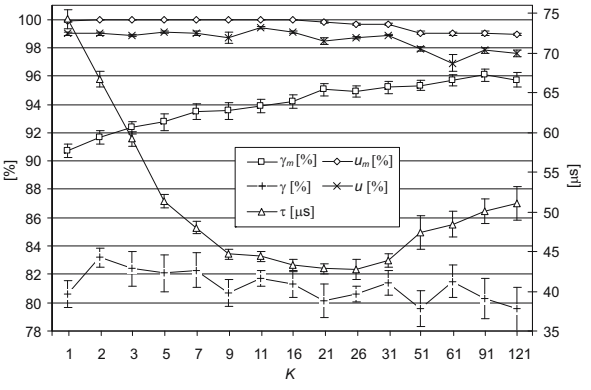

Fig. 1. Time of Sigma-if network output signal generation τ, the classification accuracies of training and test data for the final (u and γ) and for the best networks obtained (u_m and γ_m) for the Sonar problem vs. the number of hidden neuron input connections groups K (network architecture: 60 inputs, 30 hidden neurons, 2 outputs).

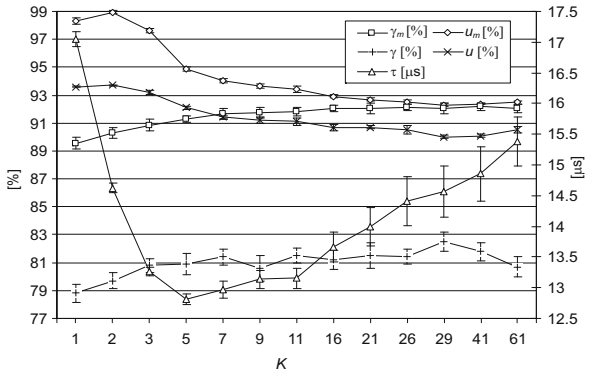

Fig. 2. Time of Sigma-if network output signal generation τ, the classification accuracies of training and test data for the final (u and γ) and for the best networks obtained (u_m and γ_m) for the HeartC problem vs. the number of hidden neuron input connections groups K (networks architecture: 28 inputs, 10 hidden neurons, 2 outputs).

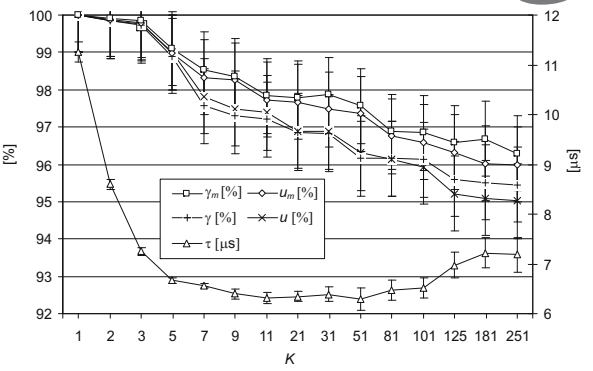

Fig. 4. Sigma-if network activity of hidden connections hca, the classification accuracy of test data for the final (γ) and the best networks obtained (γ_m) for the Mushroom problem vs. the number of hidden neuron input connections groups K (network architecture: 125 inputs, 2 hidden neurons, 2 outputs).

ber of Sigma-if neuron input groups K to more than one results in an increase in generalization γ and classification accuracy of test data γ_m of the best networks obtained during trainings. At the same time one can observe a simultaneous decrease in the overall data processing time τ. The drawback here is a decrease in the classification accuracy of training data u also visible in the case of the best generated networks (parameter u_m). Typical examples of such dependencies, for small and medium size benchmark problems such as Wine, Votes, Crx or Wisconsin Breast Cancer, can be observed for the Sonar and HeartC problems, which are presented in Figs. 1 and 2 (as the number of input connections groups K is discrete, values in the presented figures are connected with lines only to ease the analysis of the results). For larger problems, e.g., Adult and Mushroom (Figs. 3 and 4), the increase in γ and γ_m for given parameters is at most small and can be observed

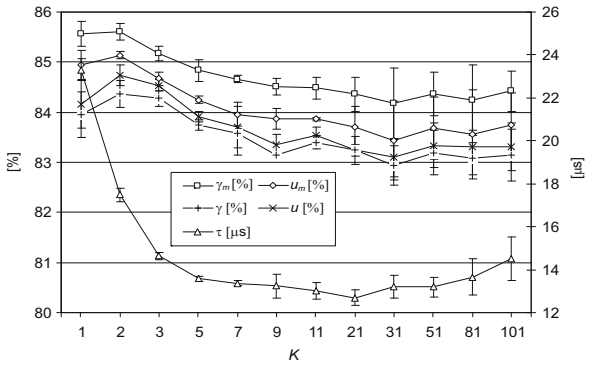

Fig. 3. Time of Sigma-if network output signal generation τ, the classification accuracies of training and test data for the final (u and γ) and for the best networks obtained (u_m and γ_m) for the Adult problem vs. the number of hidden neuron input connections groups K (networks architecture: 105 inputs, 4 hidden neurons, 2 outputs).

only for numbers of input groups K less than five. The observed decrease in the classification accuracy of training data is most probably caused by the fact that it is harder to learn when the neuron's input space is changed every ω epoch. It should also be remembered that, especially for neural networks with a larger number of inputs, a low value of the aggregation threshold φ^* can have significant influence on the network performance both on training and test data, indirectly setting a strong limit on the number of network inputs being processed for greater values of K. In the case of the Mushroom data set for $K = 11$, the increase of φ^* from 0.6 to 1.8 resulted in an increase of the average values of u, γ and γ_m to $99.3\pm0.9\%$, $99.4\pm0.7\%$ and $99.8\pm0.7\%$, respectively, while average values of the activity of hidden connections hca, the activity of network inputs nia and the number of inputs niu used were still as low as $11 \pm 4\%$, $20 \pm 7\%$ and $52 \pm 9\%$ (cf. Figs. 4 and 9). Thus by tuning parameters of the Sigma-if network to the problem size one can achieve very good results also for big data sets.

However, and more importantly, for all benchmark problems considered, the obtained increase in the classification accuracy of test data (γ and γ_m) is a result of rejecting redundant or noisy signals from processed data and the consequence of the reduction of problem complexity by decreasing its dimensionality. Another source of such properties of the Sigma-if network is splitting the initial problem into a set of K subproblems due to multi-step, conditional generation of neurons outputs. In turn, a decrease in the network's outputs generation time τ is caused by reduction of the network's hidden connections activity hca (see Fig. 5 for Sonar and Fig. 6 for Votes problems).

It is also worth noting that the visible increase in the HeartC data processing time τ, for K greater than seven inputs groups, is the effect of a linear increase in the time cost connected with the existence of additional instruc-

tions for processing the grouping vector $\overline{\theta}$. This factor can be easily seen for all benchmark problems considered for the numbers of groups K greater than the given number of network inputs. Without it, the data processing time would semi logarithmically decrease with rising K. This reflects the character of the changes of Sigma-if network hidden hca and input connection activities nia as a function of K, which can be observed for the Votes problem in Figs. 6 and 8. Therefore the obtained strong reduction of hidden connections activities confirms earlier conclusion that the data processing time reduction is connected with Sigma-if neurons' selective attention abilities, which can be observed also for large problems in Figs. 7 and 9 (Adult and Mushroom problem, respectively). All this is clear evidence that Sigma-if neurons use selective attention, and that this can reduce the generalization error level as well as data processing costs.

The conducted experiments disclose also that for a Sigma-if model generated with the use of the presented training method, selective attention can be observed on the level of the whole Sigma-if network. The analysis of results indicates (Figs. 7–9) that, when a significant decrease in network input activity nia occurs, one can expect a simultaneous reduction in the number (niu) of Sigma-if network inputs used to classify data, without a notable decrease in classification accuracy in comparison to the analogous multilayer feedforward network with sigmoidal neurons.

The above results form strong evidence that the presented generalized delta rule for the Sigma-if neural network can be effectively used to generate valuable classification models with selective attention functionality. But it is also interesting how such a method influences the length of the training process. As can be seen in Fig. 10 for chosen benchmark classification problems such as Sonar, Crx, Wine, HeartC and Breast Cancer Wisconsin, the training

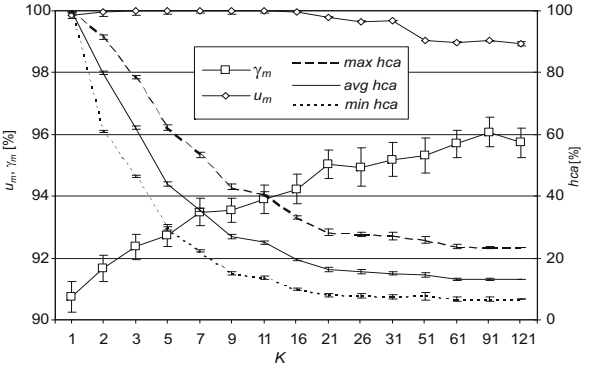

Fig. 5. Sigma-if network activity of hidden connections hca, the classification accuracy of training u_m and test γ_m data for the best networks obtained for the Sonar problem vs. the number of hidden neuron input connections groups K (network architecture: 60 inputs, 30 hidden neurons, 2 outputs).

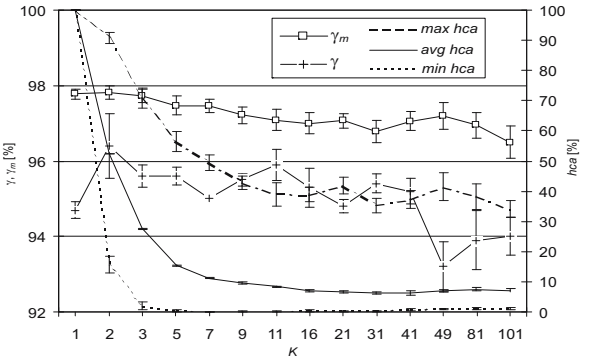

Fig. 6. Sigma-if network activity of hidden connections hca, the classification accuracy of test data for the final (γ) and the best networks obtained (γ_m) for the Votes problem vs. the number of hidden neuron input connections groups K (network architecture: 48 inputs, 2 hidden neurons, 2 outputs).

of the Sigma-if neural network (for K greater than 1 and less than 9) takes 20–25% fewer training epochs than the training of the MLP network ($K = 1$). In connection with observed 25–40% reduction of the computation time for the Sigma-if network outputs, this can accelerate the training process even more than twice.

6. Summary and future work

In this work the generalized delta rule for the Sigma-if neural network was formally presented. Its detailed derivation was shown on the basis of an analogous derivation for the MLP network. For completeness, the backpropagation algorithm combined with the self-consistency idea was discussed, as the training method which can use the derived equation to train Sigma-if neural networks.

In the second part of this article, results of experiments that demonstrate the usability of the derived

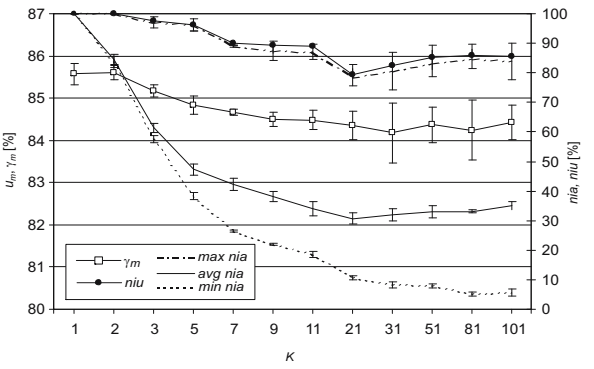

Fig. 7. Sigma-if network inputs activity nia, the number of network inputs used niu, the classification accuracy of training u_m and test γ_m data for the best networks obtained for the Adult problem vs. the number of hidden neuron input connections groups K (network architecture: 105 inputs, 4 hidden neurons, 2 outputs).

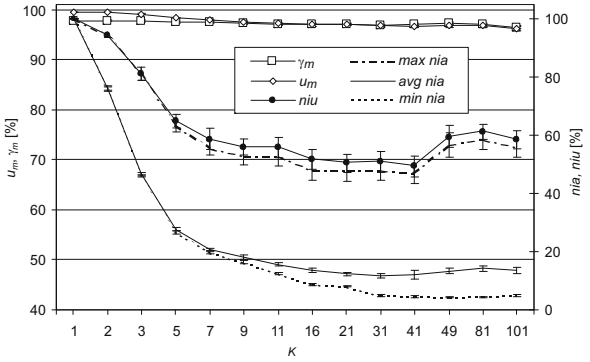

Fig. 8. Sigma-if network inputs activity nia, the number of network inputs used niu, the classification accuracy of training u_m and test γ_m data for the best networks obtained for the Votes problem vs. the number of hidden neuron input connections groups K (network architecture: 48 inputs, 2 hidden neurons, 2 outputs).

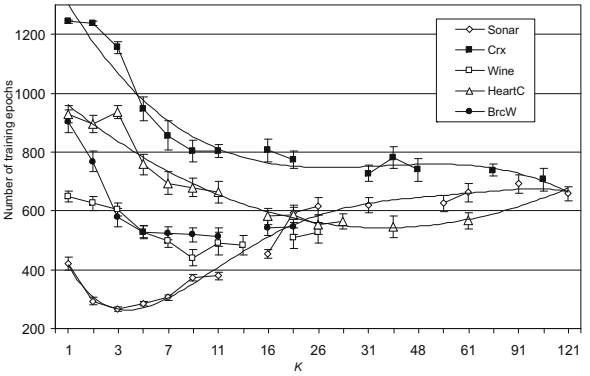

Fig. 10. Number of training epochs of the backpropagation algorithm for the Sigma-if network for selected UCI Machine Learning Repository problems vs. the number of hidden neuron input connections groups K.

equation were shown. For selected classification benchmark problems of the UCI Machine Learning Repository, trained Sigma-if networks were able to achieve better classification results than the best MLP networks. But what is more important, the obtained Sigma-if neural networks possessed also the selective attention ability. It was shown how it increases neural network classification properties and how it reduces the time of data processing by the network. The resulting effect of training epochs number reduction was also discussed.

While the Sigma-if network has no specialized or separate attention guiding unit, all observed attentional activities can emerge only as an effect of synergy between individual neurons. Thus the Sigma-if model accompanied with the presented training method can be a very promising solution for applications such as remote sensing in dispersed sensor networks as well as automatic robot navigation and control. This is because the selective attention feature introduces new possibilities in the area of analyzing the network decision process via its inputs activity interpretation. This can point at features of given data sets that are most important for classification, and help to identify features that are irrelevant, redundant or contaminated by noise. All this makes the Sigma-if neural network a very useful tool in the data acquisition and analysis domain.

Due to very interesting theoretical and practical properties, the Sigma-if model and the presented training method should be further tested on benchmark and real-life data. Also, the whole idea of synchronized conditional signals aggregation should be further explored, as aggregation functions other than the one considered in this work can be proposed, and for many of them derivation of the generalized delta rule can be challenging. Preliminary experiments show that there exist at least a few of such aggregation functions which allow achieving even better results than presented in this work.

Another issue worth exploring is examination if the Sigma-if network could be successfully trained with use of fast converging methods such as Broyden–Fletcher–Goldfarb–Shanno and Levenberg–Marquardt. Those methods use local approximates of Hessian matrix of the neural network error function, which can fail as the Sigma-if network in each training step operates potentially in a different subspace of initial set of parameters. All this makes a wide and promising direction of research on neuronal models of low-level selective attention, and is presently a subject of continuous investigation.

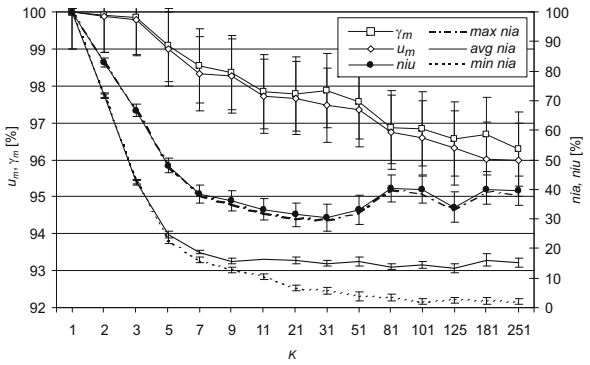

Fig. 9. Sigma-if network inputs activity nia, the number of network inputs used niu, the classification accuracy of training u_m and test γ_m data for the best networks obtained for the Mushroom problem vs. the number of hidden neuron input connections groups K (network architecture: 125 inputs, 2 hidden neurons, 2 outputs).

References

Broadbent, D. (1982). Task combination and selective intake of information, *Acta Psychologica* **50**(3): 253–290.

Desimone, R. and Duncan, J. (1995). Neural mechanisms of selective visual-attention, *Annual Review of Neuroscience* **18**(1): 193–222.

Duch, W. and Jankowski, N. (1999). Survey of neural transfer functions, *Neural Computing Surveys* **2**(1): 163–212.

Durbin, R. and Rumelhart, D. (1990). Product units: A computationally powerful and biologically plausible extension to backpropagation networks, *Neural Computation* **1**(1): 133–142.

Feldman, J. and Ballard, D. (1982). Connectionist models and their properties, *Cognitive Science* **6**(3): 205–254.

Ferguene, F. and Toumi, F.F. (2009). Dynamic external force feedback loop control of a robot manipulator using a neural compensator—Application to the trajectory following in an unknown environment, *International Journal of Applied Mathematics and Computer Science* **19**(1): 113–126, DOI: 10.2478/v10006-009-0011-9.

Fonseca, L., Jimenez, J., Leburton, J. and Martin, R. (1998). Self-consistent calculation of the electronic structure and electron-electron interaction in self-assembled InAs-GaAs quantum dot structures, *Physical Review B* **57**(7): 4017–4026.

Gupta, M. (2008). Correlative type higher-order neural units with applications, *IEEE International Conference on Automation and Logistics, ICAL 2008, Qingdao, China*, pp. 715–718.

Hager, G. and Toyama, K. (1999). Incremental focus of attention for robust visual tracking, *International Journal of Computer Vision* **35**(1): 45–63.

Houghton, G. and Tipper, S. (1996). Inhibitory mechanisms of neural and cognitive control: Applications to selective attention and sequential action, *Brain and Cognition* **30**(1): 20–43.

Huk, M. (2004). The sigma-if neural network as a method of dynamic selection of decision subspaces for medical reasoning systems, *Journal of Medical Informatics & Technologies* **7**(1): 65–73.

Huk, M. (2006). Sigma-if neural network as a use of selective attention technique in classification and knowledge discovery problems solving, *Annales UMCS Informatica AI* **5**(2): 121–131.

Huk, M. (2009). Learning distributed selective attention strategies with the Sigma-if neural network, *in* M. Akbar and D. Hussain (Eds.), *Advances in Computer Science and IT*, In-Tech, Vukovar, pp. 209–232.

Indiveri, G. (2008). Neuromorphic VLSI models of selective attention: From single chip vision sensors to multi-chip systems, *Sensors* **8**(9): 5352–5375.

Korbicz, J., Obuchowicz, A. and Uciński, D. (1994). Unidirectional networks, *in* L. Bolc (Ed.), *Artificial Neural Networks: Foundations and Applications*, Akademicka Oficyna Wydawnicza PLJ, Warsaw, pp. 35–58.

Körding, K. and König, P. (2001). Neurons with two sites of synaptic integration learn invariant representations, *Neural Computation* **13**(12): 2823–2849.

Mel, B. (1990). The sigma-pi column: A model of associative learning in cerebral cortex, *Technical report*, CNS Memo 6, Computation and Neural Systems Program, California Institute of Technology, Pasadena, CA.

Mel, B. (1992). The clusteron: Toward a simple abstraction for a complex neuron, *in* J. Moody, S. Hanson and R. Lippmann (Eds.), *Advances in Neural Information Processing Systems*, Vol. 4, Morgan Kaufmann, San Mateo, CA, pp. 35–42.

Neville, R. and Eldridge, S. (2002). Transformations of sigma-pi nets: Obtaining reflected functions by reflecting weight matrices, *Neural Networks* **15**(3): 375–393.

Niebur, E., Hsiao, S. and Johnson, K. (2002). Synchrony: A neuronal mechanism for attentional selection?, *Current Opinion in Neurobiology* **12**(2): 190–194.

Noh, T., Song, P. and Sievers, A. (1991). Self-consistency conditions for the effective-medium approximation in composite materials, *Physical Review B* **44**(11): 5459–5464.

Noton, D. and Stark, L. (1971). Scanpaths in saccadic eye movements while viewing and recognizing patterns, *Vision Research* **11**(9): 929–942.

Olshausen, B., Anderson, C. and Van Essen, D. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information, *The Journal of Neuroscience* **13**(11): 4700–4719.

Pedro, J. O. and Dahunsi, O.A. (2011). Neural network based feedback linearization control of a servo-hydraulic vehicle suspension system, *International Journal of Applied Mathematics and Computer Science* **21**(1): 137–147, DOI: 10.2478/v10006-011-0010-5.

Raczkowski, D., Canning, A. and Wang, L. (2001). Thomas-fermi charge mixing for obtaining self-consistency in density functional calculations, *Physical Review B* **64**(12): 121101–121105.

Rumelhart, D., Hinton, G. and McClelland, J. (1986). A general framework for parallel distributed processing, *in* D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, Vol. 1, The MIT Press, Cambridge, MA, pp. 45–76.

Stark, L., Privitera, C. and Azzariti, M. (2000). Locating regions-of-interest for the mars rover expedition, *International Journal of Remote Sensing* **21**(17): 3327–3347.

Treisman, A. (1960). Contextual cues in selective listening, *Quarterly Journal of Experimental Psychology* **12**(4): 242–248.

Tsotsos, J., Culhane, S. and Cutzu, F. (2001). From foundational principles to a hierarchical selection circuit for attention, *in* J. Braun, C. Koch and J. Davis (Eds.), *Visual Attention and Cortical Circuits*, MIT Press, Cambridge, MA, pp. 285–306.

Vanrullen, R. and Koch, C. (2003). Visual selective behavior can be triggered by a feed-forward process, *Journal of Cognitive Neuroscience* **15**(2): 209–217.

Weber, C. and Wermter, S. (2007). A self-organizing map of sigma-pi units, *Neurocomputing* **70**(13–15): 2552–2560.

Maciej Huk works at the Institute of Informatics of the Wrocław University of Technology, Poland. He received the M.Sc. degree in 2001 and the Ph.D. in 2007, both in computer science. His current research interests within the scope of artificial intelligence are the theory and applications of artificial neural networks in selective attention systems, efficient crossover operators for genetic algorithms and multiple classifier systems. He also works on distributed sensor networks and contextual data analysis. He is the coordinator of the *Selective attention in data analysis* research group within the Polish Cluster on Knowledge and Innovation Community for Information and Communication Technologies. Currently he also works as a software architect for the Gigaset software development center.

Received: 22 November 2010
Revised: 14 June 2011
Re-revised: 19 October 2011