Theor Chim Acta (1993) 86: 257–267

Theoretica Chimica Acta

Application of high-order neural networks in chemistry

Vladimír Kvasnička, Štěpán Sklenák, and Jiří Pospíchal

Department of Mathematics, Slovak Technical University, 81237 Bratislava, Slovak Republic

Received June 5, 1992/Accepted December 7, 1992

Summary. The effectiveness and usefulness of the so-called high-order neural networks for classification of chemical objects is demonstrated. The high-order neural networks usually do not need hidden neurons for correct interpretation of patterns. A simple formula for partial derivatives of the minimized objective (error) function is derived, which is used for production of weight coefficients during the adaptation process. An illustrative example dealing with inductive and resonance effects of functional groups by the second-order neural network is presented.

Key words: Neural network – Inductive effects – Resonance effects – Molecular descriptors

1 Introduction

The paradigm of neural networks [1–3] offers new mathematical tools equipped with learning features, which are able to classify patterns of quite diverse nature. In particular, applications of neural networks in chemistry [4, 5] are well illustrated by effective correlations between molecular structure and activity. Neural networks can be also used like "expert systems" that are able to classify molecules by their structural fragments or special properties. Usually, these applications are carried out by standard feed-forward neural networks adapted by the back-propagation method [6].

The purpose of the present communication is to demonstrate an extension of the neural network approach towards the so-called high-order neural networks. These neural networks are able to capture higher-order correlations between neurons and thus they usually offer considerably enhanced performance in adaptation and generalization processes. Moreover, knowledge determining "expert rules" can be deduced crafting networks to reflect complex informational structures.

Minsky and Papert [7] studied two-layer neural networks (perceptrons) of all orders and they concluded that high-order perceptrons are impractical due to the exponential explosion of the number of high-order weight coefficients, and that the first-order perceptrons are limited only to linearly separable patterns. Since most problems of real interest are not linearly separable, this is very serious limitation of two layer first-order neural networks. A way how to overcome this limitation is to introduce additional layers that are composed of the so-called hidden neurons [6]. The principal meaning of hidden neurons consists in the fact that they create an internal representation of input patterns, which is linearly separable by output neurons. Such an extension of the neural-network approach involves an adaptation rule for the hidden neurons, which was successfully accomplished by Rumelhart et al. [6] by the so-called back-propagation method. Unfortunately, this adaptation method for hidden neurons is often very slow, it requires thousands of iterations to converge and sometimes does not converge at all, due to the local minimum problem. The above problems can be surmounted by using high-order terms [6–9], which are to some extent equivalent to previously specified hidden neurons. In an ideal case, all hidden neurons may be substituted by considering the high-order weight coefficients. Then all patterns from the training set are correctly classified by a simple two-layer neural network that contains these high-order terms.

The effectiveness and usefulness of high-order neural networks will be illustrated by their application for classification and prediction of inductive and resonance effects (represented by sigma constants) of functional groups. These parameters, initially introduced in physical organic chemistry [10], offer very important tools to describe an influence of functional groups on reactivity of synthons [11–13] (reaction cores). The input information to the neural network should properly represent the topology and basic physical parameters of the functional groups. So we chose simple descriptors [14, 15] as frequencies of appearance of special labeled rooted subgraphs in the functional groups. As a consequence, the functional groups are described by graph-theoretical parameters and the calculation of additional physical or physico-chemical parameters of functional groups is avoided. The obtained results are very encouraging and support common chemical beliefs that the properties of molecular systems are mainly determined by structural formulae. Moreover, the fact that the application was successfully performed entirely by two-layer neural networks is potentially very encouraging for further applications of neural networks to the classification and prediction of the chemical reactivity problem based purely on simple molecular descriptors reflecting in some "additive" manner the structural formulae.

2 Theory

A feed-forward neural network [3] may be formally determined as an oriented graph [16], $G = (V, E)$, where the set $V = \{v_1, v_2, \ldots, v_N\}$ is composed of N neurons and the set $E = \{e_1, e_2, \ldots, e_M\}$ is composed of M connections. Each connection $e \in E$ is interpreted as an ordered pair of neurons from V, $e = [v, v']$; we say that the connection e is outgoing from the neuron v and incoming to the neuron v'. The subset $\Gamma(v) \subseteq V$ is composed of all neurons (called the successors of v) that are adjacent to v by connections outgoing from v. Analogously, $\Gamma^{-1}(v) \subseteq V$ is composed of all neurons (called the predecessors of v) that are adjacent to v by connections incoming to v. The neuron set V is divided into three disjoint subsets, $V = V_I \cup V_H \cup V_0$, where the subsets are called input, hidden, and output, respectively. The input neurons are incident only with outgoing connections, $V_I = [\forall v \in V\colon |\Gamma(v)| > 0$ and $|\Gamma^{-1}(v)| = 0\}$, the hidden neurons are incident at least with one incoming connection and one outgoing connection, $V_H = \{\forall v \in V: |\Gamma(v)| > 0 \text{ and } |\Gamma^{-1}(v)| > 0\}$, and finally, the output neurons are incident at least with one incoming connection, $V_0 = \{\forall v \in V: |\Gamma^{-1}(v)| > 0\}$. For standard (first-order) neural networks the activities of hidden and output neurons are determined by:

$$x_i = f(\xi_i), \tag{1a}$$

$$\xi_i = \vartheta_i + \sum_j \omega_j^i x_j, \tag{1b}$$

where the summation runs over all indices j from the set $\Gamma^{-1}(i)$, i.e. over all predecessors of the neuron indexed by i. The entries ϑ_i and ω_j^i are threshold and weight coefficients assigned to the vertex v_i and the connection $[v_j, v_i]$, respectively. The transfer function $f(\xi)$ is a monotonously increasing function that fulfills the asymptotic conditions $f(\xi) \to B$ as $\xi \to \infty$ and $f(\xi) \to A$ as $\xi \to -\infty$, where $A \leqslant 0 < B$. For instance, these requirements are simply met if the transfer function is:

$$f(\xi) = \frac{B + A\exp(-\xi)}{1 + \exp(-\xi)}, \tag{2a}$$

with the first derivative determined by:

$$f'(\xi) = \frac{[f(\xi) - A][f(\xi) - B]}{A - B}. \tag{2b}$$

Most frequently, the transfer function is usually applied either for $A = 0$, $B = 1$ or for $A = -1$, $B = 1$. The first case corresponds to classical sigmoidal function and the second case corresponds to an analog of hyperbolic tangent function.

A possibility of how to generalize the concept of feed-forward neural networks is to enlarge the basic formula of Eq. (1) so that its argument will account for not only constant (threshold coefficient) and linear terms (sum of weighted activities) but also quadratic, cubic, ... terms. This means that the activities are determined as followed [9].

$$x_i = f(\xi_i), \tag{3a}$$

$$\xi_i = \vartheta_i + \sum_j \omega_j^i x_j + \sum_{j \leqslant k} \omega_{jk}^i x_j x_k + \cdots, \tag{3b}$$

where the second summation runs over all pairs $j, k \in \Gamma^{-1}(i)$ restricted by $j \leqslant k$, etc. The entities $\omega_{jk}^i, \ldots$ are weight coefficients assigned to pairs of connections that are incoming to the same neuron. The term ω_α^i will denote a weight coefficient assigned to connections that are incoming to the neuron i and outgoing from neurons with the composite index $\alpha = (j, k, \ldots)$ constrained by $j \leqslant k \leqslant \cdots$. If the above formula contains at most quadratic terms, then the network is called second-order; the highest order terms in (3b) determines the order of a neural network.

A supervised adaptation process [3] of high-order neural networks entails looking for threshold and weight coefficients that give for a pair of prescribed input and output activity vectors x_I and $\hat{x}_0$ an output vector $x_0 = F(x_I)$, where F is a mapping realized by the neural network, and the vector x_0 should be "closely" related to the prescribed (required) output activities vector $\hat{x}_0$. Let us introduce an objective function:

$$E=\tfrac{1}{2}(x_0-\hat{x}_0)^2=\tfrac{1}{2}\sum_k g_k^2, \tag{4a}$$

$$g_k=\begin{cases}(x_k-\hat{x}_k), & (\text{for } k\in V_0)\\ 0 \quad , & (\text{for } k\notin V_0)\end{cases} \tag{4b}$$

where x_k and $\hat{x}_k$ are entries of x_0 and $\hat{x}_0$, respectively. A goal of adaptation processes is to find threshold and weight coefficients that will minimize the objective function E.

This minimization may be carried out by a gradient method [17], e.g., most simply by the steepest descent method [17, 3]. For the implementation of this method we have to know all partial derivatives $\partial E/\partial\vartheta_i$ and $\partial E/\partial\omega_\alpha^i$ of the objective function with respect to the threshold and weight coefficients. These partial derivatives may be expressed as:

$$\frac{\partial E}{\partial\vartheta_i}=\frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial\vartheta_i}=\frac{\partial E}{\partial x_i}f'(\xi_i), \tag{5a}$$

$$\frac{\partial E}{\partial\omega_\alpha^i}=\frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial\omega_\alpha^i}=\frac{\partial E}{\partial x_i}f'(\xi_i)\prod_{j\in\alpha}x_j. \tag{5b}$$

Comparing these two equations we arrive at the basic relationship between partial derivatives $\partial E/\partial\vartheta_i$ and $\partial E/\partial\omega_\alpha^i$:

$$\frac{\partial E}{\partial\omega_\alpha^i}=\frac{\partial E}{\partial\vartheta_i}\prod_{j\in\alpha}x_j. \tag{6}$$

Then the whole process of calculation of partial derivatives $\partial E/\partial\vartheta_i$ and $\partial E/\partial\omega_\alpha^i$ may be reduced to the substantially simpler calculation of $\partial E/\partial\vartheta_i$. Two-, three-, . . . index derivatives $\partial E/\partial\omega_\alpha^i$ are determined by one-index derivatives $\partial E/\partial\vartheta_i$ and a product of activities x_j with indices taken from the composite index α. The partial derivatives $\partial E/\partial x_i$ from the right-hand side of (5) are expressed as follows:

$$\begin{aligned}\frac{\partial E}{\partial x_i}&=\frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial x_i}+\sum_{l\in\Gamma(i)}\frac{\partial E}{\partial x_l}\frac{\partial x_l}{\partial x_i}\\ &=g_i+\sum_{l\in\Gamma(i)}\frac{\partial E}{\partial x_l}f'(\xi_l)\frac{\partial\xi_l}{\partial x_i}\\ &=g_i+\sum_{l\in\Gamma(i)}\frac{\partial E}{\partial\vartheta_l}\frac{\partial\xi_l}{\partial x_i}\\ &=g_i+\sum_{l\in\Gamma(i)}\frac{\partial E}{\partial\vartheta_l}\left(\omega_i^1+\sum_{k\geqslant i}\omega_{ik}^1x_k+\sum_{k\leqslant i}\omega_{ki}^1x_k+\cdots\right),\end{aligned} \tag{7}$$

where, for simplicity, we have displayed only the terms up to the second order. The higher-order terms are slightly more complex. Introducing this result in Eq. (5a) we arrive at the final formula for partial derivatives $\partial E/\partial\vartheta_i$:

$$\frac{\partial E}{\partial\vartheta_i}=f'(\xi_i)\left[g_i+\sum_{l\in\Gamma(i)}\frac{\partial E}{\partial\vartheta_l}\left(\omega_i^1+\sum_{k\geqslant 1}\omega_{ik}^1x_k+\sum_{k\leqslant i}\omega_{ki}^1x_k+\cdots\right)\right], \tag{8}$$

for $i\in V_H\cup V_0$. For the feed-forward neural networks Eq. (8) allows a simple recurrent calculation (called back-propagation method [3]) of the partial derivatives $\partial E/\partial\vartheta_i$. This calculation starts for output neurons that are incident merely with incoming lines (for the feed-forward neural networks there exists at least one such a neuron), we put $\partial E/\partial \vartheta_i = f'(\xi_i)g_i$. Continuing this process for neurons that are predecessors of the neurons considered in the previous step, the partial derivatives $\partial E/\partial \vartheta_i$ are simply calculated by Eq. (8). Finally, the weight coefficients ω^i_α are calculated according to Eq. (6).

The above method of calculation of partial derivatives may be simply generalized for more than one pair of input-output vectors x_I and $\hat{x}_0$:

$$x_I^{(1)}/\hat{x}_0^{(1)}, x_I^{(2)}/\hat{x}_0^{(2)}, \ldots, x_I^{(r)}/\hat{x}_0^{(r)}, \tag{9}$$

which form the so-called *training set*. The objective function is then determined by:

$$E = \sum_{i=1}^{r} E^{(i)}, \tag{10a}$$

$$E^{(i)} = \tfrac{1}{2}(x_0^{(i)} - \hat{x}_0^{(i)})^2, \tag{10b}$$

where $x_0^{(i)}$ is the output vector of the neural network determined by (6) as a response to an input vector $x_I^{(i)}$, and $\hat{x}_0^{(i)}$ are required output vectors assigned to input vectors $x_I^{(i)}$. Partial derivatives of the generalized objective function are then equal to the sum of the partial derivatives of $E^{(i)}$ evaluated by Eqs. (6) and (8).

If we know the gradient of the objective function, then the adaptation process of a neural network is realized by a minimization of the objective function of Eq. (10) with respect to threshold and weight coefficients. The steepest-descent minimization method accelerated by the so-called momentum method [3] is based on the following updating of threshold and weight coefficients:

$$\omega_\alpha^{i(k+1)} = \omega_\alpha^{i(k)} - \lambda \frac{\partial E}{\partial \omega^i_\alpha} + \mu\, \Delta\omega_\alpha^{i(k)} \tag{11a}$$

$$\vartheta_i^{(k+1)} = \vartheta_i^{(k)} - \lambda \frac{\partial E}{\partial \vartheta_i} + \mu\, \Delta\vartheta_i^{(k)} \tag{11b}$$

where the positive parameters $\lambda > 0$ should be both sufficiently small to ensure the convergence of the adaptation process and sufficiently large to achieve fast convergence. The momentum parameter μ is taken from the semiopen interval $(0, 1]$, usually $\mu = 0.7$–0.9. Finally, the terms $\Delta\omega_\alpha^{i(k)}$ and $\Delta\vartheta_i^{(k)}$ correspond to the previous changes of weight and threshold coefficients, respectively.

3 Simple illustrative example

We have described an adaptation process, which could, in principle, determine the threshold and weight coefficients that minimize the objective function E of the higher-order neural network. For simplicity we restrict ourselves to the third-order neural network in which only "off-diagonal" weight coefficients $\omega^i_{jkm}, j < k < m$, are considered. For input patterns determined by 0–1 digits the "diagonal" terms ω^i_{jjj} are irrelevant according to the property $x^3 = x$, for $x = 0$ or $x = 1$. These third-order neural networks will be illustrated by a simple example of summation of three 1-digit binary numbers. In the example, when the first-order neural networks are used, the hidden neurons should be used [3] to achieve the proper classification of input patterns.

3.1 Summation of three 1-digit binary numbers

Let us study an application of the third-order neural network approach to the summation of three 1-digit binary numbers, schematically:

$$x_1 + x_2 + x_3 = x_4 x_5, \tag{12}$$

where x_i's are binary 0–1 numbers. All possible eight different cases are displayed in the following table:

No.	x_1	x_2	x_3	x_4	x_5	Meaning
1	0	0	0	0	0	$0+0+0=0$
2	0	0	1	0	1	$0+0+1=1$
3	0	1	0	0	1	$0+1+0=1$
4	0	1	1	1	0	$0+1+1=2$
5	1	0	0	0	1	$1+0+0=1$
6	1	0	1	1	0	$1+0+1=2$
7	1	1	0	1	0	$1+1+0=2$
8	1	1	1	1	1	$1+1+1=3$

(13)

The considered neural network is composed of three input neurons (with activities x_1, x_2, x_3), two output neurons (with activities x_4, x_5), and no hidden neurons. All input and output neurons are connected by oriented connections outgoing from input neurons and incoming to output neurons. According to the commutation rule of all entries in the summation of Eq. (12) the input neurons should be symmetric. This means that the weight coefficients should be restricted by:

$$\omega_1^i = \omega_2^i = \omega_3^i, \qquad \omega_{12}^i = \omega_{13}^i = \omega_{23}^i, \tag{14}$$

for $i = 4, 5$. The potential ξ_i assigned to an output neuron v_i is determined in the framework of third-order neural-network approach as:

$$\xi_i(x_1, x_2, x_3) = \vartheta_i + \omega_1^i(x_1 + x_2 + x_3) + \omega_{12}^i(x_1x_2 + x_1x_3 + x_2x_3) \\ + \omega_{123}^i x_1x_2x_3. \tag{15}$$

If the input activities are taken from the table (13), then we arrive at two systems of inequalities, which determine the threshold and weight coefficients of output neurons:

$$\begin{aligned} \xi_4(0,0,0) &= \vartheta_4 && < 0, \\ \xi_4(1,0,0) &= \vartheta_4 + \omega_1^4 && < 0, \\ \xi_4(1,1,0) &= \vartheta_4 + 2\omega_1^4 + \omega_{12}^4 && > 0, \\ \xi_4(1,1,1) &= \vartheta_4 + 3\omega_1^4 + 3\omega_{12}^4 + \omega_{123}^4 && > 0, \end{aligned} \tag{16a}$$

and

$$\begin{aligned} \xi_5(0,0,0) &= \vartheta_5 && < 0, \\ \xi_5(1,0,0) &= \vartheta_5 + \omega_1^5 && > 0, \\ \xi_5(1,1,0) &= \vartheta_5 + 2\omega_1^5 + \omega_{12}^5 && < 0, \\ \xi_5(1,1,1) &= \vartheta_5 + 3\omega_1^5 + 3\omega_{12}^5 + \omega_{123}^5 && > 0. \end{aligned} \tag{16b}$$

The right-hand sides of these inequalities may be formally identified with auxiliary positive constants (i.e. the inequalities (16a) are rewritten as $\vartheta_4 = -\varepsilon_0$,

$\vartheta_5 + \omega_1^5 = -\varepsilon_1$, $\vartheta_4 + 2\omega_1^4 + \omega_{12}^4 = \varepsilon_{12}$, $\vartheta_4 + 3\omega_1^4 + 3\omega_{12}^4 + \omega_{123}^4 = \varepsilon_{123}$, and similarly for (16b)), solving these equations we get:

$$\begin{aligned} \vartheta_4 &= -\varepsilon_0, \\ \omega_1^4 &= \varepsilon_0 - \varepsilon_1, \\ \omega_{12}^4 &= -\varepsilon_0 + 2\varepsilon_1 + \varepsilon_{12}, \\ \omega_{123}^4 &= \varepsilon_0 - 3\varepsilon_1 - 3\varepsilon_{12} + \varepsilon_{123}, \end{aligned} \tag{17a}$$

and

$$\begin{aligned} \vartheta_5 &= -\varepsilon_0, \\ \omega_1^5 &= \varepsilon_0 + \varepsilon_1, \\ \omega_{12}^5 &= -\varepsilon_0 - 2\varepsilon_1 - \varepsilon_{12}, \\ \omega_{123}^5 &= \varepsilon_0 + 3\varepsilon_1 + 3\varepsilon_{12} + \varepsilon_{123}. \end{aligned} \tag{17b}$$

If the third-order weight factor ω_{123} is neglected, then the system of inequalities (16a) is satisfied for ε's constants restricted by $\varepsilon_1 + \varepsilon_{12} > \varepsilon_0/3$, but then the last inequality of the system (16b) is not fulfilled. Its left-hand side provides $\xi_5(1, 1, 1) = -\varepsilon_0 - 3\varepsilon_1 - 3\varepsilon_{12} > 0$, this condition is never fulfilled for positive ε's constants. In summay: a two-layer neural network, which simulates the summation of Eq. (12) of three 1-digit binary numbers, must be at least of third-order, its lower-order versions are unable to simulate the considered problem. However, the necessity of third-order neural network is not a rule. The second-order two-layer neural network is often able to do correct classification without a necessity to apply a hidden-neuron approach even for patterns that are not linearly separable. This could be proved for example by successful simulation of logical function XOR [9].

4 Classification of inductive and resonance sigma constants by the second-order neural networks

The inductive and resonance sigma constants σ_I and σ_R are very important parameters of physical organic chemistry [10] that characterize electronic properties of functional groups. They offer a quantitative description of qualitative conceptions used in organic chemistry [18] when the influence of functional groups on a molecular skeleton (e.g. benzene ring) is considered. Quantum-chemical approaches [19] to understand the physical nature of these constants are not very successful. Probably, the main reason for this is the fact that theoretical models take into account the environmental effects in an adequate way. At the beginning of the eighties effective procedures for the quantification of inductive and resonance effects were developed by Gasteiger et al. [20–22].

Functional groups are formally considered as rooted molecular graphs [16] in which the vertices (atoms) and edges (bonds) are represented by additional symbols that properly specify their physical nature. The root of these molecular graphs corresponds to the atom immediately attached to the parent molecule. We shall restrict ourselves to univalent and electroneutral functional groups, i.e. they are attached to molecules by single bonds and are not viewed to be a holder of net positive or negative charge. We have used a modification of the approach initially suggested by Johnson et al. [14], which was successfully used by the present authors for prediction of meta products of nitration [15] and ^{13}C-NMR chemical shifts [23] in a series of monosubstituted benzenes by a neural network method. Recently, the same approach has been used [24] for the prediction and classification of a large set composed of 87 functional groups from the point of view of their experimental inductive and resonance sigma constants. The used feed-forward neural networks are three-layer, where the second (or middle) layer is composed of eight hidden neurons.

The fourteen atom and bond descriptors of functional groups are specified in Table 1 [14, 15]. These descriptors characterize the univalent functional groups as inputs of neural networks. In particular, an entry of the descriptor is equal to a nonnegative integer counting the appearances of the given property in atoms of 1st (2nd, 3rd) level in the structural formula of the functional group, see Tables 1 and 2. The term "level" of atom is determined by the smallest number of bonds that are placed between an actual atom of substituent and the parental molecule.

We have selected from Exner's review article [25] 37 substituents and their experimental inductive and resonance sigma constants, this set of functional groups is divided into a disjoint training set (31 functional groups) and a testing set (6 functional groups), see Table 3. The choice of functional groups was restricted to univalent and electroneutral groups. In the testing set there are given "representative" choices of different kinds of those substituents.

Table 1. Descriptors of functional groups

d_i	Meaning of the descriptor
	First-level descriptors
d_1	Number of lone electron pairs on the first level atom
d_2	Sum of the main quantum numbers (each decreased by one) for the valence shell of the first level atom
d_3	Number of hydrogen atoms attached to the first level atom
d_4	Number of phenyl groups attached to the parental molecule
	Second-level descriptors
d_5	Number of lone electron pairs on the second level atoms
d_6	Sum of the main quantum numbers (each decreased by one) for the valence shell of the second level atoms
d_7	Number of hydrogen atoms attached to the second level atoms
d_8	Number of phenyl groups attached to the first level atoms
d_9	Number of pi bonds that connect the first and second level atoms
	Third-level descriptors
d_{10}	Number of lone electron pairs on the third level atoms
d_{11}	Sum of the main quantum numbers (each decreased by one) for the valence shell of the third level atoms
d_{12}	Number of hydrogen atoms attached to the third level atoms
d_{13}	Number of phenyl groups attached to the second level atoms
d_{14}	Number of pi bonds that connect the second and third level atoms

Table 2. Descriptors of illustrative examples

—X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
$—NO_2$	0	1	0	0	4	2	0	0	2	0	0	0	0	0
$—CH_2Ph$	0	1	2	0	0	0	0	1	0	0	0	0	0	0
$—SO_2Ph$	0	2	0	0	4	2	0	1	2	0	0	0	0	0
$NHCOCH_3$	1	1	1	0	0	1	0	0	0	2	2	3	0	1

Table 3. Experimental and currently estimated sigma constants

No.	—X	exp.		est.	
		σ_I	σ_R	σ_I	σ_R
Training set					
1	—H	0.00	0.00	0.14	0.01
2	$—CH_3$	0.00	−0.11	0.01	−0.10
3	$—CH_2CH_3$	0.00	−0.10	0.01	−0.10
4	—Ph	0.12	−0.11	0.05	−0.11
5	$—CH_2Ph$	0.03	−0.12	0.03	−0.12
6	$—CH_2CN$	0.20	−0.09	0.20	−0.10
7	$—CH_2OH$	0.11	−0.05	0.08	−0.03
8	$—CH_2Cl$	0.17	0.00	0.18	−0.01
9	$—CF_3$	0.40	0.08	0.40	0.08
10	$—CCl_3$	0.36	0.00	0.36	0.00
11	—CN	0.57	0.13	0.53	0.09
12	—CHO	0.25	0.24	0.28	0.28
13	$—COCH_3$	0.30	0.16	0.29	0.18
14	$—CONH_2$	0.28	0.14	0.30	0.12
15	$—NH_2$	0.17	−0.48	0.10	−0.49
16	$—N(CH_3)_2$	0.17	−0.52	0.17	−0.53
17	$—NHCOCH_3$	0.28	−0.25	0.28	−0.24
18	—NHCOPh	0.28	−0.25	0.28	−0.26
19	$—NO_2$	0.67	0.15	0.66	0.15
20	—OH	0.24	−0.62	0.28	−0.54
21	$—OCH_3$	0.30	−0.45	0.31	−0.45
22	$—OCF_3$	0.55	−0.19	0.55	−0.19
23	—SH	0.27	−0.19	0.31	−0.24
24	$—SCOCH_3$	0.39	0.08	0.39	0.10
25	—SCN	0.56	−0.90	0.56	−0.87
26	$—SOCH_3$	0.49	0.00	0.49	0.00
27	—F	0.54	−0.34	0.51	−0.31
28	—Cl	0.47	−0.23	0.46	−0.24
29	—N=NPh	0.19	0.06	0.20	0.07
30	—OPh	0.40	−0.34	0.39	−0.35
31	$—OCOCH_3$	0.38	−0.23	0.38	−0.26
Testing set					
1	—Br	0.47	−0.19	0.40	−0.17
2	—I	0.40	−0.16	0.33	−0.10
3	$—CH(CH_3)_2$	0.00	−0.12	0.05	−0.27
4	—COPh	0.27	0.19	0.31	0.23
5	—COOH	0.30	0.14	0.39	0.06
6	$—SCH_3$	0.30	−0.20	0.34	−0.31

The used second-order neural network is composed of two layers. The juxtaposed input and output layers are fully connected by oriented connections. The input layer is composed of fourteen neurons that correspond to descriptors (input activities). The output layer contains two neurons, whose activities correspond to inductive and resonance sigma constants. Since the sigma constants are all between -1 and 1, the constants A and B from transfer function Eq. (2) can reasonably be fixed as $A = -1$ and $B = 1$. The adaptation process based on Eqs. (6) and (8) is carried out for $\lambda = 0.01$ and $\mu = 0.8$, then after 1000 iterations we have achieved $E = 0.01$ and $|\text{grad } E| = 10^{-4}$, where E is determined by Eq. (8a–b). The sigma constants for substituents from the training as well as the testing set as predicted by adapted neural network are given in Table 3.

5 Discussion

We have demonstrated that the higher-order neural networks are able to classify input patterns correctly without a necessity to use the hidden-neuron approach. This fact might be of great importance for the simple interpretation and better understanding of results produced by higher-order neural networks. Their applications to chemically interesting problems, represented in this communication by classification of inductive and resonance sigma constants, seem to be very promising for the construction of algorithms, which classify and predict chemical reactivity of organic molecules from their simple structural descriptors. These descriptors offer a simple possibility of how to code the molecular structural formulae in way useful for the prediction of chemical reactivity by a neural network. It seems that such an approach of coding molecular structure combined with the higher-code neural networks gives simple algorithmic tools for the construction of computer systems. Those systems seem able to predict qualitatively (or semiquantitatively) the properties of molecular systems – in particular those properties that are strictly localized on the fragment (subgraph) of whole structural formula.

Neural networks are analytically very complicated algorithmic tools, therefore their properties according to their design, used transfer and objective functions are usually derived from numerical experiments. The number of "parameters" which should be fitted is rather too large to carry out exhausting search for best choice. These unfortunate facts generally lead authors in chemical applications [4] to make choice of design of neural networks according to their experience and intuition rather than by exact approaches. Numerical problems connected with neural network applications in chemistry, especially in structure-property relationships, have been recently studied by Maggiora et al. [26].

References

1. Simpson PK (1990) Artificial neural systems. Pergamon Press, NY
2. Wassermann P (1989) Neural computing: Theory and practice. Van Nostrand Reinhold, Princeton
3. Rumelhart DE, McClelland JL (eds) (1986) Parallel distributed processing, Vols I and II. MIT Press, Cambridge, Mass

4. Župan J, Gasteiger J (1991) J Anal Chem 248:1
5. Tetrahedron Computer Methodology (1990) 3(1)
6. Rumelhart DE, Hinton GE, Williams RJ (1986) in ref. [3] Vol I, p 318
7. Minsky M, Papert S (1969) Preceptrons. An introduction to computational geometry. MIT Press, Cambridge, Mass
8. Owechko Y, Dunning GJ, Maron E, Soffer BH (1987) Appl Opt 26:1900
9. Giles CL, Maxwell T (1987) Appl Opt 26:4872
10. Hammett LP (1970) Physical organic chemistry. McGraw-Hill, NY
11. Corey EJ (1967) Pure Appl Chem 14:19
12. Kvasnička V, Pospíchal J (1990) Int J Quant Chem 28:235
13. Koča J, Kratochvíl M, Kvasnička V, Matyska L, Pospíchal J (1989) Synthon model of organic chemistry and synthesis design. Springer Verlag, Berlin
14. Zou Y, Johnson M, Tsai CC (1990) J Chem Inf Comput Sci 30:442
15. Kvasnička V, Pospíchal J (1991) J Mol Structure (THEOCHEM) 235:227
16. Harary F (1969) Graph theory. Addison-Wesley, Reading, Mass
17. Polak E (1971) Computational methods in optimization. Academic Press, NY
18. Ingold CK (1969) Structure and mechanism in organic chemistry. Cornell Univ Press, Ithaca
19. Hehre WJ, Taft RW, Topson RD (1976) Progr Phys Org Chem 12:159
20. Gasteiger J, Marsili M (1980) Tetrahedron 36:3219
21. Hutchings MG, Gasteiger J (1983) Tetrahedron Lett 24:2451
22. Gasteiger J, Saller H (1985) Angew Chem Intern Ed Engl 24:687
23. Kvasnička V, Sklenák Š, Pospíchal J (1992) J Mol Struct (THEOCHEM) 277:87
24. Kvasnička V, Sklenák Š, Pospíchal J (1993) J Am Chem Soc 115:1495
25. Exner O (1987) in: Chapman NB, Shorter J (eds) Correlation analysis in chemistry. Plenum Press, NY, p 439
26. Maggiora GM, Elrod DW, Trenary RG (1992) J Chem Inf Comput Sci 32:732