Quantitative Structure-Activity Relationships of Benzodiazepines by Recursive Cascade Correlation

Anna Maria Bianucci, Alessio Micheli, Alessandro Sperduti, Antonina Starita

Abstract— **An application of Recursive Cascade Correlation to the Quantitative Structure-Activity Relationships (QSAR) of a class of Benzodiazepines is presented. Recursive Cascade Correlation is a neural network model recently proposed for the processing of structured data. This allows the direct treatment of the chemical compounds as labeled ordered trees, which constitutes a novel approach to QSAR. Our approach compares favorably versus the traditional QSAR treatment based on equations.**

Keywords— **Cascade Correlation for Structures, Recursive Neural Networks, QSAR, Benzodiazepines.**

I. Introduction

THE aim of the present work consists in proposing a new approach to the analysis of the Quantitative Structure-Activity Relationships (QSAR), based on recursive neural networks [1]. The earliest attempts to find relationships between molecular properties of biologically active compounds and their activities were performed since the past century. A systematic approach to the treatment of these relationships was mainly introduced by C. Hansh et al. in the 60s [2] with the development of equations able to correlate the biological activity to physical and chemical properties of biologically active compounds. Several different models were then developed based on equations exploiting a wide variety of molecular properties, including structural descriptors such as topological indexes [3]. The importance of the QSAR studies relays in the fact that it enables us to design new drugs on the basis of the known structure-activity relationships supplied by the QSAR analysis. This capability strongly impacts in drug discovery allowing a more effective use of the resources.

Neural networks have been applied with different modalities to QSAR [4], [5]. The most typical approach is the one where a feedforward neural network is trained on physicochemical parameters [6], [7], [8]. In addition, structural information coded through topological indexes or vectorial graph codes may be given as input to the feedforward network [9], [10], [11]. Finally, a template based approach has also been explored [12].

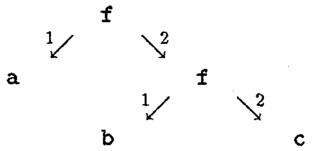
The present work is based on an attempt of directly correlate the biological activity to the molecular structure by using recursive neural networks which are capable to directly process molecular structures represented as labeled directed ordered acyclic graphs. The specificity of the proposed approach stems from the ability of recursive networks to automatically encode the structural information depending on the computational problem at hand, i.e., the representation of the molecular structures is not defined a priori, but learned on the basis of the training set. The specific model we study in this paper is Recursive Cascade Correlation [13], a generalization of Recurrent Cascade Correlation [14] to the processing of structures. The experimental results obtained on a class of Benzodiazepines confirms that recursive neural networks are very promising candidates for successful QSAR applications.

The paper is organized as follows. In Section II we recall some preliminary notions on graphs. Then a brief description of recursive neural networks and a more detailed description of Recursive Cascade Correlation is given in Section III. The QSAR task is introduced in Section IV, where the representation of the molecular structures is briefly explained. Experimental results are presented in Section V and conclusions are drawn in Section VI.

II. Preliminaries on Graphs

Here we consider structured domains which are sets of labeled directed ordered acyclic graphs (DOAGs). For a DOAG we mean a DAG $\boldsymbol{Y}$ with vertex set vert$(\boldsymbol{Y})$ and edge set egd$(\boldsymbol{Y})$, where for each vertex $v \in$ vert$(\boldsymbol{Y})$ a total order on the edges leaving from v is defined. Labels are tuples of variables and are attached to vertices. The void DOAG will be denoted by the special symbol ξ.

For example, in the case of graphs representing logical terms, the order on outgoing edges is immediately induced by the order of the arguments to a function; e.g., the logical term `f(a,f(b,c))` can be represented as:



We shall require the DOAG either to be empty or to possess a supersource, i.e. a vertex $s \in$ vert$(\boldsymbol{Y})$ such that every vertex in vert$(\boldsymbol{Y})$ can be reached by a directed path starting from s. Note that if a DOAG does not possess a supersource, it is still possible to define a convention for adding an extra vertex s (with a minimal number of outgoing edges), such that s is a supersource for the expanded

Anna Maria Bianucci is with Dipartimento di Scienze Farmaceutiche, Via Bonanno 6, 56126, Pisa, Italy; E-mail: bianucci@farm.unipi.it. Alessio Micheli, Alessandro Sperduti, and Antonina Starita are with Dipartimento di Informatica, Corso Italia 40, 56125 Pisa, Italy; E-mail: {micheli,perso,starita}@di.unipi.it.

0-7803-4859-1/98 $10.00©1998 IEEE

DOAG [1]. The function *source*($\boldsymbol{Y}$) returns the (unique) supersource of $\boldsymbol{Y}$.

The *indegree* of a node v is the cardinality of the set of outgoing edges from v, while the *outdegree* of v is the cardinality of the set of edges incident on v. In the following, a generic class of DOAGs with labels in $\mathcal{I}$ and bounded (but unspecified) indegree and outdegree, will be denoted by $\mathcal{I}^{\#}$.

III. Recursive neural networks

Recursive neural networks [1] are neural networks able to perform mappings from a set of labeled graphs to the set of real vectors. Specifically, the class of functions which can be realized by a recursive neural network can be characterized as the class of functional graph transductions $\mathcal{T} : \mathcal{I}^{\#} \rightarrow I\!\!R^k$, where $\mathcal{I} = I\!\!R^n$, which can be represented in the following form

$$\mathcal{T} = g \circ \hat{\tau}, \tag{1}$$

where $\hat{\tau} : \mathcal{I}^{\#} \rightarrow I\!\!R^m$ is the *encoding* (or *state transition*) function and $g : I\!\!R^m \rightarrow I\!\!R^k$ is the *output* function. Specifically, given a DOAG $\boldsymbol{Y}$, $\hat{\tau}$ is defined recursively as

$$\hat{\tau}(\boldsymbol{Y}) = \begin{cases} \mathbf{o} \text{ (the null vector in } I\!\!R^m) & \text{if } \boldsymbol{Y} = \xi \\ \tau(s, \boldsymbol{Y}_s, \hat{\tau}(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}(\boldsymbol{Y}^{(o)})) & \text{otherwise} \end{cases} \tag{2}$$

where the *c-model* function τ is defined as

$$\tau : V \times I\!\!R^n \times \underbrace{I\!\!R^m \times \cdots \times I\!\!R^m}_{\text{o times}} \rightarrow I\!\!R^m \tag{3}$$

where V is the set of all vertices, $I\!\!R^n$ denotes the label space, while the remaining domains represent the encoded subgraphs spaces up to the maximum outdegree of the input domain $\mathcal{I}^{\#}$, o is the maximum outdegree of DOAGs in $\mathcal{I}^{\#}$, $s = source(\boldsymbol{Y})$, $\boldsymbol{Y}_s$ is the label attached to the supersource of $\boldsymbol{Y}$, and $\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(o)}$ are the subgraphs pointed by s. The function τ is called c-model function since it defines a computational model for the encoding function.

Note that, because of eq. (2), $\mathcal{T}$ is *causal* since τ only depends on the current node and nodes descending by it. Moreover, when τ does not depend on any specific vertex, i.e., $\tau(\boldsymbol{Y}_s, \hat{\tau}(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}(\boldsymbol{Y}^{(o)}))$, then $\mathcal{T}$ is also *stationary*. In this paper we focus on stationary transductions.

Example III.1 (Encoding of logical terms)
Given a stationary encoding function $\hat{\tau}$, the encoding of the logical term `f(a,f(b,c))` is defined by the following set of equations

$$\hat{\tau}\left(\begin{array}{ccccccc} & & \mathtt{f} & & & & \\ & {}^{1}\!\swarrow & & \searrow^{2} & & & \\ \mathtt{a} & & & & \mathtt{f} & & \\ & & & {}^{1}\!\swarrow & & \searrow^{2} & \\ & & \mathtt{b} & & & & \mathtt{c} \end{array}\right) =$$

$$\tau(\mathtt{f}, \hat{\tau}(\mathtt{a}), \hat{\tau}\left(\begin{array}{ccccc} & & \mathtt{f} & & \\ & {}^{1}\!\swarrow & & \searrow^{2} & \\ \mathtt{b} & & & & \mathtt{c} \end{array}\right)),$$

$$\hat{\tau}\left(\begin{array}{ccccc} & & \mathtt{f} & & \\ & {}^{1}\!\swarrow & & \searrow^{2} & \\ \mathtt{b} & & & & \mathtt{c} \end{array}\right) = \tau(\mathtt{f}, \hat{\tau}(\mathtt{b}), \hat{\tau}(\mathtt{c})),$$

$$\hat{\tau}(\mathtt{b}) = \tau(\mathtt{b}, nil, nil),$$

$$\hat{\tau}(\mathtt{a}) = \tau(\mathtt{a}, nil, nil),$$

$$\hat{\tau}(\mathtt{c}) = \tau(\mathtt{c}, nil, nil),$$

where a, b, and c denote the graphs with a single node labeled a, b, and c, respectively.

Concerning the output function g, it can be defined as a map

$$g : I\!\!R^m \rightarrow I\!\!R^k. \tag{4}$$

Note that eqs. (3) and (4) only describe the general form for τ and g. Different realizations can be given which satisfy the above equations. For example, both τ and g can be implemented by feedforward neural networks. Here, however, we consider the following neural realizations for τ:

$$\tau(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)}) = \boldsymbol{F}(\boldsymbol{W}\boldsymbol{l} + \sum_{j=1}^{o} \widehat{\boldsymbol{W}}_j \boldsymbol{x}^{(j)} + \boldsymbol{\theta}), \tag{5}$$

where $\boldsymbol{F}_i(\boldsymbol{v}) = f(v_i)$ (sigmoidal function), $\boldsymbol{l} \in I\!\!R^n$ is a label, $\boldsymbol{\theta} \in I\!\!R^m$ is the bias vector, $\boldsymbol{W} \in I\!\!R^{m \times n}$ is the weight matrix associated with the label space, $\boldsymbol{x}^{(j)} \in I\!\!R^m$ are the vectorial codes obtained by the application of the encoding function $\hat{\tau}$ to the subgraphs $\boldsymbol{Y}^{(j)}$, and $\widehat{\boldsymbol{W}}_j \in I\!\!R^{m \times m}$ is the weight matrix associated with the jth subgraph space.

Concerning the output function $g(\cdot)$, it can be defined as a set of standard neurons taking in input the encoded representation $\boldsymbol{x}$ of the graph, i.e.,

$$g(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{M}\boldsymbol{x} + \boldsymbol{\beta}), \tag{6}$$

where $\boldsymbol{M} \in I\!\!R^{k \times m}$ and $\boldsymbol{\beta} \in I\!\!R^k$ are the weight matrix and bias terms defining $g(\cdot)$, respectively.

A. Learning with Recursive Cascade Correlation

In this section we discuss how a neural graph transduction $\mathcal{T}$ can be learned using an extension of the Cascade Correlation algorithm. The standard Cascade-Correlation algorithm [15] creates a neural network using an incremental approach for the classification (or regression) of unstructured patterns. The starting network $\mathcal{N}_0$ is a network without hidden nodes trained by a Least Mean Square algorithm. If network $\mathcal{N}_0$ is not able to solve the problem, a hidden unit u_1 is added such that the *correlation* between the output of the unit and the residual error of network $\mathcal{N}_0$ is maximised[1]. The weights of u_1 are frozen and the remaining weights are retrained. If the obtained network $\mathcal{N}_1$ cannot solve the problem, new hidden units are added which are connected (with frozen weights) with all the inputs and previously installed hidden units. The resulting

[1] Since the maximization of the correlation is obtained using a gradient ascent technique on a surface with several maxima, a pool of hidden units is trained and the best one selected.

network is a *cascade* of nodes. Fahlman extended the algorithm to the classification of sequences, obtaining good results [14].

In the following, we show that the Cascade-Correlation can be further extended to structures by using our computational scheme. In fact, the shape of the c-model function can be expressed component-wise by the following set of equations:

$$\begin{aligned}
\tau_1 &= h_1(\boldsymbol{l}, \hat{\tau}_1(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}_1(\boldsymbol{Y}^{(o)})), \\
\tau_2 &= h_2(\boldsymbol{l}, \hat{\tau}_1(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}_1(\boldsymbol{Y}^{(o)}), \\
&\quad \hat{\tau}_2(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}_2(\boldsymbol{Y}^{(o)}), \hat{\tau}_1(\boldsymbol{Y})), \\
&\;\vdots \\
\tau_m &= h_m(\boldsymbol{l}, \hat{\tau}_1(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}_1(\boldsymbol{Y}^{(o)}), \\
&\quad \hat{\tau}_2(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}_2(\boldsymbol{Y}^{(o)}), \ldots, \\
&\quad \hat{\tau}_m(\boldsymbol{Y}^{(1)}), \ldots, \hat{\tau}_m(\boldsymbol{Y}^{(o)}), \\
&\quad \hat{\tau}_1(\boldsymbol{Y}), \ldots, \hat{\tau}_{m-1}(\boldsymbol{Y})),
\end{aligned} \tag{7}$$

where the h_i are suited nonlinear functions of the arguments.

Specifically, the output of the kth hidden unit, in our framework, can be computed as

$$\begin{aligned}
\tau_k(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)}) = f(\sum_{i=0}^{n} w_i^{(k)} l_i + \sum_{v=1}^{k} \sum_{j=1}^{o} \hat{w}_{(v,j)}^{(k)} \boldsymbol{x}_v^{(j)} + \\
+ \sum_{q=1}^{k-1} \bar{w}_q^{(k)} \tau_q(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)})),
\end{aligned} \tag{8}$$

where $w_{(v,j)}^{(k)}$ is the weight of the kth hidden unit associated with the output of the vth hidden unit computed on the jth subgraph code $\boldsymbol{x}^{(j)}$, and $\bar{w}_q^{(k)}$ is the weight of the connection from the qth (frozen) hidden unit, $q < k$, and the kth hidden unit. The output of the network (with k inserted hidden units) is then computed according to equation (6), where $\boldsymbol{M} \in \mathbb{R}^k$ since we have a single output unit. Moreover, since we are interested in biological activity prediction, the output unit is set to be linear, i.e., $g(\boldsymbol{x}) = \boldsymbol{M}^t \boldsymbol{x} + \beta$.

Learning is performed as in standard Cascade Correlation by interleaving the minimization of the total error function (LMS) and the maximization of the correlation of the new inserted hidden unit with the residual error. The main difference with respect to standard Cascade Correlation is in the calculation of the derivatives. According to equation (8), the derivatives of $\tau_k(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)})$ with respect to the weights are computed as

$$\frac{\partial \tau_k(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)})}{\partial w_i^{(k)}} = f'(l_i + \sum_{j=1}^{o} \hat{w}_{(k,j)}^{(k)} \frac{\partial \boldsymbol{x}_k^{(j)}}{\partial w_i^{(k)}}) \tag{9}$$

$$\begin{aligned}
\frac{\partial \tau_k(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)})}{\partial \bar{w}_q^{(k)}} &= f'(\tau_q(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)}) + \\
&\quad + \sum_{j=1}^{o} \hat{w}_{(k,j)}^{(k)} \frac{\partial \boldsymbol{x}_k^{(j)}}{\partial \bar{w}_q^{(k)}})
\end{aligned} \tag{10}$$

$$\begin{aligned}
\frac{\partial \tau_k(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(o)})}{\partial \hat{w}_{(v,t)}^{(k)}} &= f'(\boldsymbol{x}_v^{(t)} + \\
&\quad + \sum_{j=1}^{o} \hat{w}_{(k,j)}^{(k)} \frac{\partial \boldsymbol{x}_k^{(j)}}{\partial \hat{w}_{(v,t)}^{(k)}})
\end{aligned} \tag{11}$$

where $i = 0, \ldots, n$, $q = 1, \ldots, (k-1)$, $v = 1, \ldots, k$, $t = 1, \ldots, o$, f' is the derivative of $f()$. The above equations are recurrent on the structures and can be computed by observing that for graphs composed by a single vertex equation (9) reduces to $\frac{\partial \boldsymbol{x}_k^{(j)}}{\partial w_i^{(k)}} = l_i f'$, and all the remaining derivatives are null. Consequently, we only need to store the output values of the unit and its derivatives for each component of a structure.

Learning for the output weights proceeds as in the standard Cascade Correlation.

IV. The task

The ability of predicting the biological activity of chemical compounds belonging to classes of therapeutical interest constitutes the major aspect of the drug design. Benzodiazepines, for example, has been extensively studied since the 70s, as this class of compounds plays the major role in the field of minor tranquilizer, and several QSAR studies have been carried out aiming at the prediction of the non-specific activity (affinity) towards the Benzodiazepine/GABA$_A$ receptor. The affinity can be expressed as the inverse of the logarithm of the drug concentration C (Mol./liter) able to give a fixed biological response[2].

As a first approach, a group of Benzodiazepines (Bz) (classical 1,4-benzodiazepin-2-ones) previously analyzed by Hansch et al. [2] through the traditional QSAR equations, was analyzed. The data set analyzed by Hansch in table 2 [2] appeared to be characterized by a good molecular diversity, and this last requirement makes it particularly significant in any kind of QSAR analysis. For this reason, we have used the same data set[3]. The total number of molecules was 72.

The analyzed molecules present a common structural aspect given by the Benzodiazepine ring (see Figure 1) and they differ each other because of a large variety of substituents at the positions showed in Figure 1.

A. Molecular structure representation

The main requirement for the use of the Recursive Cascade Correlation network consists in finding a representation of molecular structures in terms of DOAGs. The candidate representation should retain the detailed information about the structure of the compound, atom types, bond multiplicity, chemical functionalities, and finally a good similarity with the representations usually adopted in chemistry.

[2] In order to characterize the fixed response, the drug concentration able to give half of the maximum response (IC$_{50}$) is commonly used.

[3] Except for the racemic compounds which are commonly recognized to introduce ambiguous information.

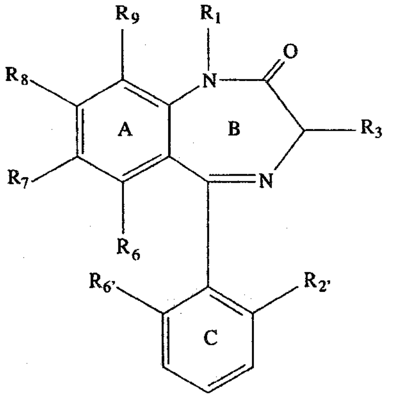


Fig. 1. The common template shared by the majority of the analyzed molecules.

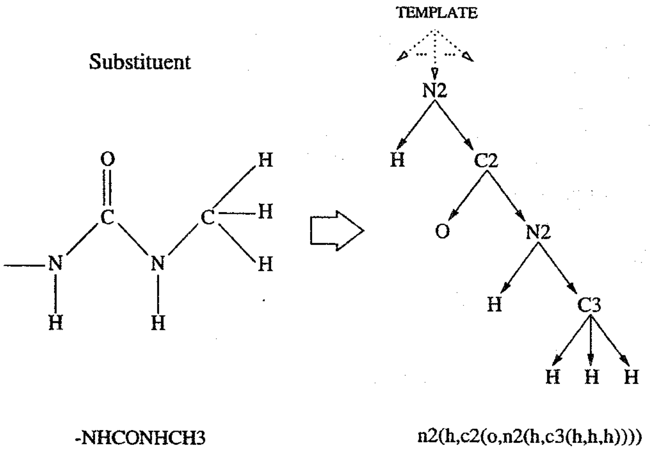


Fig. 2. Example of representation of a substituent.

An appropriate description of the molecular structures analyzed in this work is based on a labeled tree representation. This representation captures most of the structural information, except for cycles. However, since cycles mainly constitute the common shared template of the compounds, it seemed reasonable to represent a cycle as a single node where the attached label carries information about its chemical nature.

The labeled tree representation of a compound is obtained by the following minimal set of rules:

1. the root of the tree represents the Bz ring;
2. the root does have as many subtrees as substituents on the Bz ring, sorted according to the order conventionally followed in chemistry;
3. each atom (or cycle) of a substituent is represented by a node, and each bound[4] by an arc; the root of the subtree representing the substituent corresponds to the atom directly connected to the common template, and the orientation of the arcs follows the increasing levels of the trees;
4. suitable labels, representing the atom type (or cycle), are associated to the root and all the nodes;
5. the total order on the subtrees of each node is hierarchically defined according to: *i)* the subtree's depth, *ii)* the number of nodes of the subtree, *iii)* the atomic weight of the subtree's root.

An example of how a substituent is represented as a labeled ordered tree is shown in Figure 2.

V. Experimental results

For the analysis of the previously described data set, three different splittings in disjoint training and test sets of the data were used (Data set I, II, and III, respectively). Specifically, the first test set (4 compounds) has been chosen as it contains the same compounds used by Hansch for the validation of his treatment. The second one (5 compounds) has been selected as it simultaneously shows a significant molecular diversity and a wide range of affinity values. Furthermore, the included compounds were selected so that substituents, already known to increase the affinity on given positions, appear in turn in place of H-atoms, which allows the decoupling of the effect of each substituent. So, a good generalization on this test set means that the network is able to capture the relevant aspects for the prediction. The third one (4 compounds) has been randomly chosen so to test the sensitivity of the network to different learning conditions.

As target output for the networks we used $1/\log(C)$ normalized into the range $[0.6, 0.9]$. Concerning the label attached to each node, a bipolar localist representation encoding the atom types has been used.

An initial set of preliminary trials were performed in order to determine an admissible range for the learning parameters. However, no effort was done to optimize these parameters. Six trials were carried out for the simulation involving each one of the different training sets. The connection weights used in each simulation were randomly set.

Due to the low number of training data and to avoid overfitting, several expedients were used. First of all, no connection between hidden units were allowed. Then the gain of the sigmoids of the hidden units were set to 0.4. Finally, an incremental strategy (*i-strategy*) on the number of training epochs was adopted for each new inserted hidden node. This was done because allowing few epochs to the first nodes decreases the probability of overfitting, by avoiding the increase of the weight values and the subsequent saturation of the units. On the other hand, lately introduced nodes, which work with small gradients due to the reduction of the residual error, take advantage from the increased number of the epochs. Learning is stopped when the maximum error for a single compound is below 0.4 (which is actually 0.04 since we have scaled the target by a factor of 10). This tollerance is largely below the minimal tollerance needed for a correct classification of active drugs.

The main statistics computed over all the simulations for the training sets are reported in Table I, where the results obtained by Hansch et al. are reported on the first row. For each data set statistics on the number of inserted hidden units are reported, as well as on the expected mean error. Moreover, the correlation coefficient (R) and the

[4] The multiplicity of the bound is implicitly encoded in the structure of the subtree.

standard deviation of error (S), as defined in regression analysis, are reported in the last two columns. Note that R and S for Recursive Cascade Correlation are obtained by averaging over the single trials. For the first data set we have reported the results obtained when using Recursive Cascade Correlation without i-strategy on the number of training epochs (nis), by using the i-strategy (is), and by an empirically tuned version of the i-strategy (tis). The results for the corresponding test sets are reported in Table II.

In Figure 3 we have plotted the output of the network versus the desired target for each splitting of the data. Each point in the graphs represents the mean expected output, together with the deviation range, as computed over the six trials.

TABLE I

Results Obtained on Training Data Set I by Hansch et. al. (first row) and on All the Training Data Sets by Recursive Cascade Correlation. Results Obtained for Different Learning Settings Are Reported for Training Data Set I. The Correlation Coefficient (R) and the Standard Deviation of Error (S) Are Reported.

Training Set	#Units
	Mean(Min-Max)
		Mean Error
		(Min-Max)
			R	S

Hansch		0.311	0.847	0.390
Data set I nis	48.0 (44-52)	0.110 (0.099-0.120)	0.99973	0.144
Data set I is	15.3 (13-17)	0.100 (0.076-0.114)	0.99978	0.130
Data set I tis	34.0 (27-38)	0.087 (0.080-0.102)	0.99982	0.117
Data set II	19.7 (18-22)	0.087 (0.072-0.105)	0.99985	0.098
Data set III	16.5 (13-20)	0.099 (0.078-0.132)	0.99976	0.131

TABLE II

Results Obtained on Test Data Set I by Hansch et. al (first row) and on All the Test Data Sets by Recursive Cascade Correlation. Results Obtained for Different Learning Settings Are Reported for Test Data Set I. The Standard Deviation of Error (S) Is Reported.

Test Set	Mean Error(Min-Max)	Max Error(Min-Max)	S
Hansch	1.250	1.750	1.294
Data set I nis	0.757 (0.703-0.810)	0.991 (0.839-1.142)	0.792
Data set I is	0.662 (0.501-0.807)	0.839 (0.661-1.088)	0.683
Data set I tis	0.546 (0.444-0.653)	0.727 (0.523-0.973)	0.579
Data set II	0.255 (0.206-0.325)	0.606 (0.433-0.712)	0.329
Data set III	0.379 (0.279-0.494)	0.746 (0.695-0.763)	0.460

VI. Conclusions

Concerning the evaluation of the performance of the model, from the comparison with the results obtained by the traditional Hansch treatment, we can observe both a strong improvement in the fitting of the molecules included in the training set and in the test set[5], even if such a comparison cannot be considered to be rigorous since we have neglected the racemic compounds. Nevertheless, the experimental results suggest a relevant improvement over traditional QSAR techniques. In fact, this has been confirmed by the good results obtained for the second test set, where the worse predicted compound is the one bearing hydrogen atoms in place of substituents which are relevant for the prediction. Moreover, the stability of the proposed model has been confirmed by the good results obtained for the third test set. Specifically, the compound which showed the maximum variance through the trials contains a substituent which never appears in the training set, which explains the uncertainty in the response.

The main features of our approach relay on the generality of the compound representations which allows the simultaneous treatment of chemically heterogeneous compounds, and on the adaptability of the neural representations according to the computational task, thus avoiding any a priori selection of molecular descriptors, such as physicochemical properties. Finally, our approach must be regarded as a major step towards a fully structural representation and treatment of the chemical compounds.

References

[1] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures", *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.

[2] D. Hadjipavlou-Litina and C. Hansch, "Quantitative structure-activity relationships of the benzodiazepines. a review and reevaluation", *Chemical Reviews*, vol. 94, no. 6, pp. 1483–1505, 1994.

[3] L. H. Hall and L. B. Kier, *Reviews in Computational Chemistry*, chapter 9, The Molecular Connectivity Chi Indexes and Kappa Shape Indexes in Structure-Property Modeling, pp. 367–422, VCH Publishers, Inc.: New York, 1991.

[4] J. Zupan and J. Gasteiger, *Neural Networks for Chemists: an introduction*, VCH Publishers, NY(USA), 1993.

[5] J. A. Burns and George M. Whitesides, "Feed-forward neural networks in chemistry: Mathematical system for classification and pattern recognition", *Chemical Reviews*, vol. 93, no. 8, pp. 2583–2601, 1993.

[6] Y. Suzuki, T. Aoyama and H. Ichikawa, "Neural networks applied to quantitative structure-activity relationships", *J. Med. Chem.*, vol. 33, pp. 2583–2590, 1990.

[7] Ajay, "A unified framework for using neural networks to build QSARs", *J. Med. Chem.*, vol. 36, pp. 3565–3571, 1993.

[8] K. L. Peterson, "Quantitative structure-activity relationships in carboquinones and benzodiazepines using counter-propagation neural networks", *J. Chem. Inf. Comput. Sci.*, vol. 35, no. 5, pp. 896–904, 1995.

[9] D. W. Elrod, G. M. Maggiora, and R. G. Trenary, "Application of neural networks in chemistry. 1. prediction of electrophilic aromatic substitution reactions", *J. Chem. Inf. Comput. Sci.*, vol. 30, pp. 447–484, 1990.

[10] D. Cherqaoui and D. Villemin, "Use of neural network to determine the boiling point of alkanes", *J. Chem. Soc. Faraday Trans.*, vol. 90, no. 1, pp. 97–102, 1994.

[11] G. M.J. West, "Predicting phosphorus NMR shifts using neural networks. 2. factors influencing the accuracy of predictions", *J. Chem. Inf. Comput. Sci.*, vol. 35, pp. 21–30, 1995.

[12] V. Kvasnička and J. Pospichal, "Application of neural networks in chemistry.prediction of product distribution of nitration in a series of monosubstituted benzenes", *J. Mol. Struct. (Theochem)*, vol. 235, pp. 227–242, 1991.

[13] A. Sperduti, D. Majidi, and A. Starita, "Extended cascade-correlation for syntactic and structural pattern recognition", in *Advances in Structural and Syntactical Pattern Recognition*, Petra Perner, Patrick Wang, and Azriel Rosenfeld, Eds., vol. 1121 of *Lecture notes in Computer Science*, pp. 90–99. Springer-Verlag, Berlin, 1996.

[14] S. E. Fahlman, "The recurrent cascade-correlation architecture", Tech. Rep. CMU-CS-91-100, Carnegie Mellon, 1991.

[15] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture", in *Advances in Neural Information Processing*

[5] It should be noted that the relative order of the expected mean values of the affinities agrees with the desired one, which is not the case with Hansch et al. treatment.

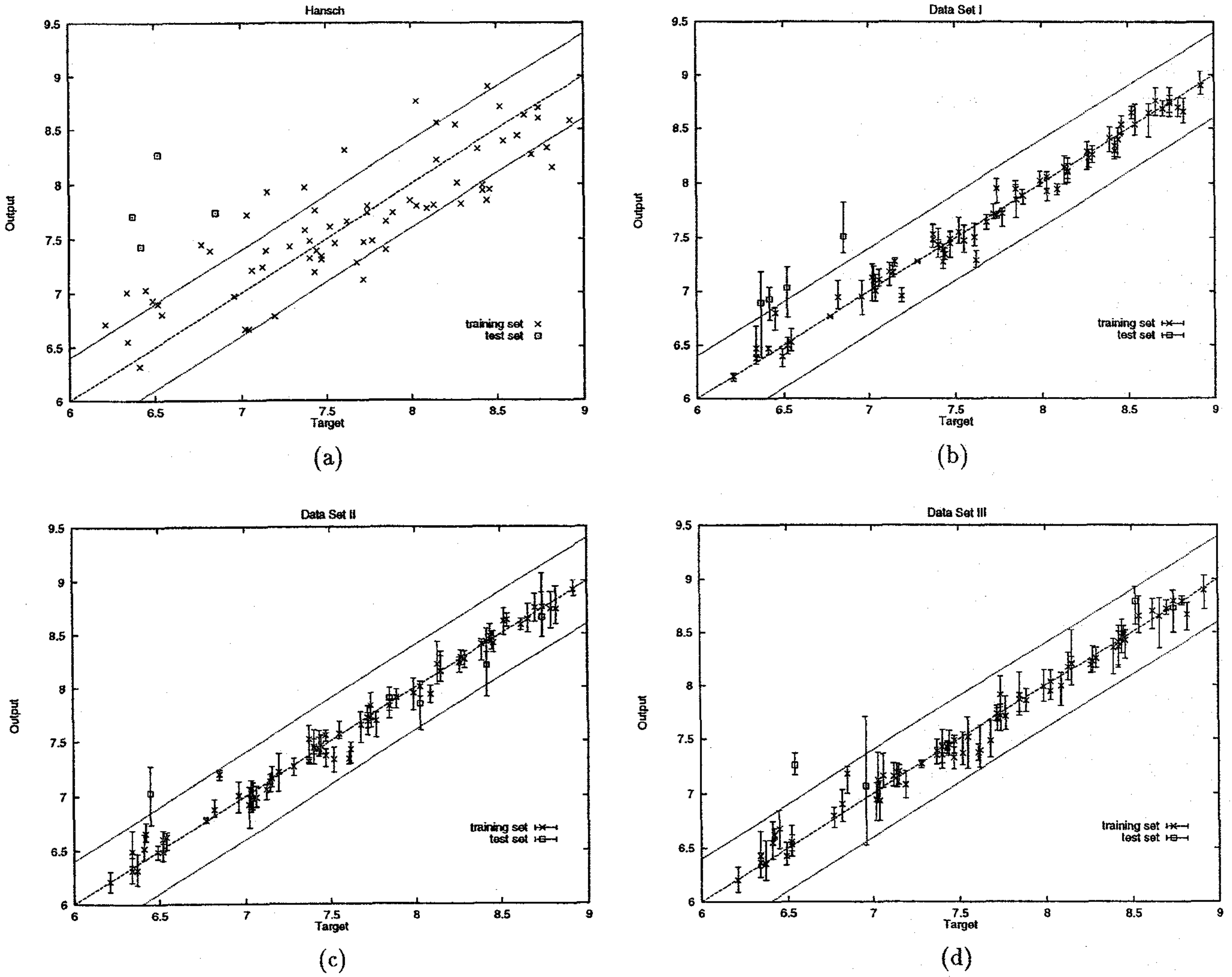


Fig. 3. Output of the Recursive Cascade Correlation network versus the desired target for each splitting of the data. Each point in the plots, except for (a), represents the mean expected output, together with the deviation range, as computed over the six trials. For the sake of presentation, both target and output values are scaled by a factor of 10. The tollerance region is shown on the plots. (a) The output computed by Hansch et al.; (b) Data set I: the test data are located at the lower left corner; (c) Data set II: the test data are spread across the input range; (d) Data set III: the large deviation range for the test compound with target close to 7 is explained by the presence of a substituent in the tested compound which does not appear in the training set.

Systems 2, D. S. Touretzky, Ed. 1990, pp. 524–532, San Mateo, CA: Morgan Kaufmann.