Toward a Principled Methodology for Neural Network Design and Performance Evaluation in QSAR. Application to the Prediction of LogP

A. F. Duprat,[†] T. Huynh,[‡] and G. Dreyfus*,[‡]

Laboratoire de Recherches Organiques and Laboratoire d'Électronique, École Supérieure de Physique et de Chimie Industrielles, 10 rue Vauquelin, 75231 Paris Cedex 05, France

Received October 26, 1997

The prediction of properties of molecules from their structure (QSAR) is basically a nonlinear regression problem. Neural networks are proven to be parsimonious universal approximators of nonlinear functions; therefore, they are excellent candidates for performing the nonlinear regression tasks involved in QSAR. However, their full potential can be exploited only in the framework of a rigorous approach. In the present paper, we describe a principled methodology for designing neural networks for QSAR and estimating their performances, and we apply this approach to the prediction of logP. We compare our results to those obtained on the same molecules by other methods.

1. INTRODUCTION

Neural networks are more and more widely used in QSAR as well as in various areas where data modeling is important. Unfortunately, the "biological" inspiration of these statistical tools too often obscures the basic issues involved in neural network design and application, in particular for QSAR applications (see ref 1 for a very valuable, lucid introductory textbook on neural nets). Therefore, the first part of the present paper is devoted to recalling briefly basic principles—some of which are not specific to neural networks—that are frequently overlooked. We insist on the fact that the sole justification of using neural networks for nonlinear regression is their parsimony. In the second part, we summarize briefly the steps to be taken in the design, training, and performance evaluation of a neural network for nonlinear regression. In the third part, we introduce a simple constructive method, based on first principles, for the selection of the variables of a neural model. Finally, we illustrate these principles by the prediction of logP; we compare the results obtained by our approach to those obtained by conventional regression techniques and demonstrate that, as expected from theoretical results, the parsimony of neural networks allows them to make a better use of the available data than polynomial regression. We also apply our model selection method and show that it allows us to effectively discriminate relevant descriptors from irrelevant ones.

2. ELEMENTS OF A PRINCIPLED APPROACH TO DATA MODELING WITH NEURAL NETWORKS

Because of their biological inspiration, neural networks are usually defined as a set of connected nonlinear elements, as shown on Figure 1. This view, however, is both useless and misleading. Neural networks, as used in QSAR, and, more generally, in data modeling applications, have nothing to do whatsoever with the way the brain works; they should be considered as just another family of parameterized nonlinear functions which, like polynomials, wavelets, Fourier series, radial basis functions, splines, etc., are nonlinear approximators;[2] some neural networks do have, however, a specific advantage over other families of parameterized functions, as will be indicated below. In the framework of statistical data modeling, which is precisely that in which neural networks are used for QSAR, these nonlinear functions are intended to approximate the regression function of the predicted property, i.e., the expectation value of the latter (viewed as a random variable) conditional to the set of variables of the model (the descriptors of the molecules in QSAR). Since the models (polynomials, neural networks, wavelets, radial functions, etc.) are parameterized functions, the goal of modeling is the following: estimate the values of the parameters of the model which best predicts the data. The difficulty of the task lies in the fact that a

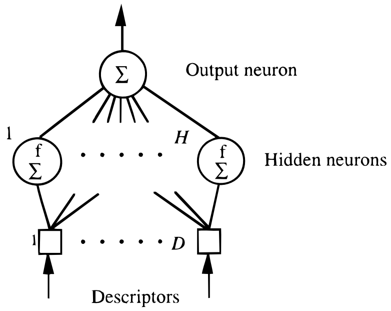

Figure 1. A multilayer perceptron (a special class of feedforward neural networks), with a layer of H "hidden" neurons and a single, linear output neuron. The output of the hidden neuron i is given by $y_i = f[\sum_{j=1}^{D} \theta_{ij} d_j]$, where $\{d_j, j = 1 \text{ to } D\}$ is the set of descriptors, $\{\theta_{ij}, j = 1 \text{ to } D)$ is a set of parameters, and where $f(.) = \tanh(.)$. The output of the network is given by $y = \sum_{k=1}^{H} \theta_k y_k$, where $\{\theta_k, k = 1 \text{ to } H\}$ is a set of parameters. The output neuron performs a linear combination of the outputs of the hidden neurons, which are nonlinear combinations of the input descriptors; adjustable weights are present in both connection layers, so that the output is nonlinear with respect to the weights of the first layer of connections. Each neuron has an additional, constant, input (usually termed "bias") which is not shown.

[†] Laboratoire de Recherches Organiques.
[‡] Laboratoire d'Électronique.

S0095-2338(98)00042-0 CCC: $15.00
Published on Web 06/04/1998

finite data base is used for estimating the parameters, whereas the number of predictions is *infinite*. It is always possible to find a model which fits the available data perfectly: it suffices to take a model with a huge number of parameters. Such overparameterized models, however, give very poor results on fresh data (data which has not been used for estimating the parameters), a phenomenon widely known in regression as *overfitting*. On the other hand, a model with too few parameters gives poor results both on the data used for estimating the parameters and on fresh data. Therefore, a good model is a tradeoff between performance on the estimation data and performance on fresh data. The appropriate approach to successful data modeling is the following: find the smallest model (i.e., the model with the smallest number of parameters), such that

•the performance on estimation data (i.e., data used for estimating the parameters) and on fresh data (i.e., data used for estimating the prediction performance) are of the same order of magnitude,

•the performances on these sets are as good as possible given the available data.

If the model is too large, the first condition is not met: the performance on estimation data is much better than the performance on fresh data; if the model is too small, performances on both sets are poor. What is meant by a "large" or a "small" number of parameters depends on the amount of available estimation data; one principle that should always be kept in mind when using *any* statistical method for estimating parameters from experimental data is the following: *the number of adjustable parameters should be small with respect to the number of data points used for estimating them*. If this simple principle is overlooked, the results, however good, are not statistically reliable. We will see below that this condition is necessary but not sufficient.

The above considerations are not specific to neural nets. What is special about neural nets and justifies the surge of interest for them in QSAR and many other areas is the fact, based on solid mathematical results,[3] that some neural networks (to be specified below) are *parsimonious approximators*, i.e., they reach a level of performance equivalent to that reached by other approximation methods *with a smaller number of parameters* or, equivalently, they give better results with the same number of parameters. Since, as mentioned above, the number of examples must be larger than the number of parameters, a parsimonious method allows the data modeler to get the same amount of information out of a smaller amount of data or equivalently to get more significant information out of a given amount of data. This result has been proved mathematically, and it is demonstrated on a QSAR example in section 5.2.2 of the present paper.

Not all neural networks are parsimonious; *neural networks whose output is nonlinear with respect to the parameters* (in addition to being nonlinear with respect to the *variables*) are more parsimonious than approximation functions which are linear with respect to the parameters, such as polynomials, Fourier series, networks of wavelets with fixed centers and dilations, networks of radial functions with fixed centers and variances, etc. The popular multilayer Perceptron (Figure 1), having one layer of hidden neurons with sigmoid activation functions and a linear output neuron, is a parsimonious neural network, since its output is nonlinear with respect to the weights of the first layer of connections. Qualitatively, the origin of parsimony is the following: when the output is linear with respect to the parameters, it is a linear combination of functions *whose shapes are fixed*; in contrast, the output of a multilayer Perceptron is a linear combination of functions *whose shapes are adjustable* through the weights of the first layer; in addition, the parameters of the linear combination (the weights of the second layer of weights) are also adjustable. The additional flexibility due to the fact that the shapes of the functions to be combined are adjusted, together with the parameters of the combination, is the key to the parsimony. Quantitatively, the property of parsimony is expressed as follows: for a given degree of accuracy, the number of weights in a multilayer Perceptron grows *linearly* with the dimensionality of the problem (i.e., with the number of descriptors in QSAR), whereas it grows *exponentially* with the dimensionality of the problem when a nonparsimonious method, such as polynomial regression, is used.

This property is not a guarantee of success with neural nets in practical applications; additional conditions are the following:

•the set of examples used for estimating the parameters (the so-called *training set*) must not only be large enough but also must be statistically representative of the data the neural network will have to predict;

•the variables input to the network must be relevant;

•the algorithm used for training the network must be efficient.

The first two conditions are definitely not specific to neural nets; they are just consequences of the fact that the estimation of parameters from experimental data is a statistical problem. How to choose the estimation data (or training set) is a problem-specific question to which no general answer can be given.

The third condition is more technical but not less important. The price to be paid for parsimony is the following: standard parameter estimation techniques, such as least squares algorithms, cannot be applied to the training of parsimonious neural nets. The estimation of the parameters of neural networks, just as the estimation of the parameters of any model, be it linear or not, neural or not, is based on the minimization of a cost function which is the sum of the squares of the differences between the measured data and the estimated data

$$J = \sum_{n=1}^{N_T} (v^n - y^n)^2$$

where v^n is the value of the measured quantity for the nth element of the training set, y^n is the estimation made by the model for the same example, and N_T is the total number of examples used for training. Since the output of the network is not linear with respect to the weights of the first layer, the cost function is not quadratic with respect to these weights; therefore, standard least squares methods cannot be used. Instead, one has to resort to gradient methods, which are iterative techniques whereby the weights are updated as a function of the gradient of the cost function with respect to the weights. The simplest way of doing this consists in updating the weights proportionally to the gradient (this method is called simple gradient descent); this is a hopelessly inefficient technique, resulting in thousands of training iterations (see for instance refs 4 and 5) without any guarantee that a minimum of the cost function (even a local one) has been reached. Much more efficient techniques, such as quasi-Newton, BFGS, or Levenberg−Marquardt algorithms[6] have been known for a long time; they cut down computation times by orders of magnitude. To the best of our knowledge, these algorithms have virtually never been used in QSAR applications (with the notable exception of ref 7). The use of an efficient training algorithm has another benefit, in addition to saving computation time: because simple gradient descent is so slow, the natural tendency is to use oversize networks, so that the algorithm has a chance to find a reasonable minimum of the cost function in a reasonable time. This, of course, contradicts the parsimony principle. By contrast, if training is performed with an efficient gradient descent method, much smaller networks may be used, so that one takes full advantage of their parsimony. [Note that there is a frequent confusion, related to training, in the literature: it is often mentioned that training is performed by *backpropagation*. Actually, backpropagation is *not* a training algorithm. Training requires two steps: (i) computation of the gradient of the cost function with respect to the weights and (ii) computation of the weight updates as a function of the gradient computed in the previous step. Because of the specific mathematical form of neural nets, the computation of the gradient may be performed, in an economical way, if the gradient is computed from the output back to the inputs (hence the term backpropagation) as a simple consequence of the formula of chained derivatives. Thus, the first of the above two steps *is* backpropagation. The second step uses the gradient, computed by backpropagation, for estimating the weight updates. This can be performed in a number of ways, the recommended way being the use of second-order methods, as mentioned above. Therefore, backpropagation is just one ingredient in a variety of training algorithms. The confusion culminates with the term "backpropagation neural network" which is frequently found (for instance in ref 8) as a substitute for "feedforward neural net" or "multilayer perceptron"; this is utterly confusing, since backpropagation can be used for any kind of neural net, including feedback (or recurrent) neural nets.]

3. DESIGNING A NEURAL NETWORK AND ESTIMATING ITS PERFORMANCES

The design and performance estimation of a neural network for QSAR requires the following ingredients:

−a set of D descriptors of molecules, which are assumed to be relevant for the prediction of the property under consideration; these descriptors are either measured or computed from molecular simulations;

−a network architecture; for application to nonlinear regression, the recommended architecture consists of a single layer of H "hidden" neurons and a single linear output neuron (Figure 1); the number of parameters of the network P is given by $P = (D+1)H + (H+1)$;

−a training algorithm, i.e., a method for estimating the values of the parameters for which the cost function

$$J = \sum_{n=1}^{N_T} (\upsilon^n - y^n)^2$$

is minimal; N_T is the number of examples, i.e., of couples $\{(d_i^n, i = 1 \text{ to } D), (\upsilon^n, n = 1 \text{ to } N_T)\}$ where υ^n is the measured value of the property to be predicted for the example n;

−a data base of examples; in the present work, several data bases were used, as explained below;

−a function for the evaluation of the performance of the neural net on fresh data (called *test set*); here we use the standard error of prediction (SEP):[4,9]

$$\text{SEP} = \sqrt{\frac{\sum_{n=1}^{N_F} (\upsilon^n - y^n)^2}{N}}$$

where N_F is the number of examples of the test set.

3.1. Descriptor Selection. The design of a set of candidate descriptors is basically a matter of insight. However, various methods have been proposed for selecting relevant descriptors among a set of candidate descriptors; most of these methods require multiple trainings of neural networks. By contrast, in the present paper, we make use of a fast method which does not rely on approximations and does not require repeated neural net trainings; it will be described in detail in section 4.

3.2. Determination of the Number of Hidden Neurons. The determination of the number of hidden neurons, hence of the number of parameters, requires a tradeoff between accuracy and parsimony, as explained above. Overfitting is the phenomenon whereby the variance of the approximation error on the training set is smaller than the noise present in the measurements; in practice, since the variance of the noise is usually unknown, overfitting is detected by the observation that the mean square error on the training set is significantly smaller than the mean square error on the test set. Therefore, the number of hidden neurons is usually found empirically, by adding neurons until overfitting is detected. This procedure was used in the present study. Recently, a rigorous statistical method for selecting the number of hidden neurons has been proposed and applied to various problems, academic and financial.[10] It is closely related to the descriptor selection method described in section 4.

3.3. Training. Since training algorithms are iterative, *overtraining* may occur: due either to an excessive number of hidden neurons or to an inappropriate distribution of examples in the training set, an overall degradation of the performance of the network may occur while the performance in the areas of input space which are well represented in the training set is improved. This problem may be alleviated by monitoring, during training, the performance of the network on a validation set (independent of the training set), which becomes minimum at some time during training, and subsequently increases while the error on the training set goes on decreasing. Thus, it is generally recommended to use three different data sets for training and evaluating the networks:

−a training set from which the gradient of the cost function and the weight updates are computed;

−a validation set, independent of the previous set, for which the cost function is computed: the network selected at the end of training is the network which gives rise to the smallest cost function on the validation set; this procedure is helpful to circumvent the above-mentioned overtraining phenomenon;[7,11−13]

−a test set, independent of the previous sets, which is used for evaluating the performance of the selected network, as described in the next section.

In the present work, the gradient of the cost function was computed by backpropagation, and the weight updates by the BFGS method. [This algorithm is implemented in the software package NeuroOne by NETRAL S.A.]

3.4. Performance Evaluation. Performance evaluation is a very important step, possibly the most costly in terms of computation time, if one wants to get a statistically meaningful estimation of the performances. It can be carried out in two ways, both of which were used in the present work:

•Several random partitions of a data base into training set, validation set, and test set may be used for training and evaluation (cross-validation, CV). The performance of a given architecture is estimated as the average of the computed SEP over all partitions.

•Alternatively, the prediction can be performed for *each example* of the data base using random partitions of the training and validation sets (leave-one-out, LoO). The overall SEP for a given network architecture is estimated by averaging the prediction errors of all the examples.

The latter method is more accurate since more examples are used for training. Conversely, the former method allows the estimation of the SEP within a much shorter time.

In the present work, additional performance evaluations were carried out on a fixed test set drawn from a different data base.

4. DESCRIPTOR SELECTION

4.1. Descriptor Selection for Nonlinear Models. Descriptor selection is often a crucial step in any data modeling problem, whether using neural nets or not. Many selection techniques have been developed for models which are linear with respect to the parameters; for models which are not linear with respect to the parameters, the problem is much more difficult.

For neural nets, several heuristics have been developed, known under the general term of “pruning” techniques. They are based on the following idea: the “saliency” of a weight of the network, i.e., the influence of a weight on the value of the cost function, can be proved to be roughly proportional to the matrix of the second derivatives of the cost function with respect to the weights, in the vicinity of a minimum of the cost function. Thus, the “pruning” techniques consist in starting with an oversize network, training it until a minimum of the cost function is reached, computing a numerical estimation of the saliency of each weight, deleting the weights with low saliency, retraining the resulting network, and carrying on the process until no improvement in the prediction performance of the network is obtained. These techniques have several drawbacks: they are computationally demanding, since they require retraining the network at each step of the procedure; in addition, they rely on the fact that the *global* minimum is reached after each training, so that, in practice, one has to train the network several times at each step of the procedure in order to have some chance of getting to the best possible minimum.

In other words, pruning methods provide an estimation of the significance of a variable which depends on the accuracy of the model; the accuracy of the model in turn depends on the initial values of the weights. In ref 14, it is proposed to circumvent the problem by (i) optimizing the initial weights through simulated annealing, a procedure which, in order to have a reasonable chance of getting to the local minimum, must be extremely slow, and which has a lot of parameters (initial configuration, annealing schedule, stopping criterion) which must be adjusted empirically and (ii) by optimizing the choice of descriptors by simulated annealing too (any other stochastic optimization method such as genetic algorithms, tabu search, probabilistic hill climbing, etc., might be used as well). The combination of these optimizations is extremely time-consuming, without any guarantee of actually getting to the optimum solution.

Therefore, for descriptor selection, it is highly preferable to use a procedure whose resulting model is guaranteed to be *unique*. Such is the case of models which are *linear with respect to the weights*. Hence, the idea of the descriptor selection method used in this work is the following: *select the descriptors on the basis of a model linear with respect to the parameters (as described in section 4.2) and subsequently use them as inputs to a parsimonious neural net*.

Thus, for variable selection, the proposed procedure capitalizes on the fact that polynomial models can be trained by least squares techniques which give a *unique* model within a very short computation time; for prediction, it capitalizes on the ability of neural nets to approximate any nonlinear regression function with a smaller number of parameters than polynomial approximation.

A somewhat similar idea was proposed in ref 7, where descriptor selection was performed with a *linear* model; this was essentially correct in the case investigated in the above reference, since the linear model was quite satisfactory, so that neural networks just provided a correction to the linear model. In general, however, this is not the case.

4.2. Descriptor Selection for a Nonlinear Model Which Is Linear with Respect to Its Parameters. A simple and efficient method for selecting the inputs of a nonlinear model which is linear with respect to its parameters was proposed by Chen et al.[15] in 1989. It is based on the Gram−Schmidt orthogonalization method, that we recall briefly below. Consider such a model, with D candidate descriptors. The output of the model is of the form

$$y = \sum_{k=1}^{K} w_k \varphi_k(x_1, x_2, ..., x_D)$$

where φ_k is a nonlinear function of the descriptors, K is the number of such functions, chosen by the designer of the model, and w_k is the weight of φ_k in the model. In polynomial regression for instance, the φ_k’s are monomials. The first step in the selection of the φ_k’s consists in ranking them in descending order of contribution to the output.

This ranking is performed as follows: we denote by N_T the number of examples, and by $\varphi_k{}^n$ the value of the kth

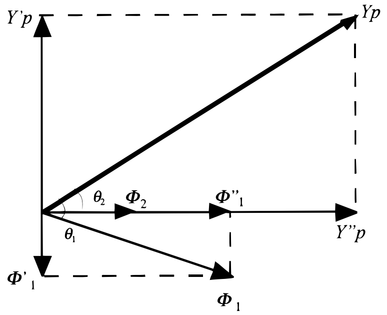

Figure 2. Illustration of the Gram−Schmidt procedure with two vectors Φ_1 and Φ_2: vector Φ_2, having the smallest angle with vector Y_p, is ranked first. Then Yp and Φ_1 are projected onto the subspace (which, here, is one-dimensional), perpendicular to the previously selected vector Φ_2, resulting in vectors Φ'_1 and Y'_p. In this particular case the procedure stops since there are only two vectors to rank.

function when the input is the vector of descriptors of example *n*; we denote by *X* the (N_T, K) matrix whose column *k* is the vector φ_k of values taken on by φ_k and by Y_p the vector, of dimension N_T, of the values of the output y_p for the examples. The function which is the most significant, i.e., which best "explains" the output, gives rise to the vector Φ_k which has the smallest angle, in N_T-dimensional space, with the output vector. This can be evaluated simply through the cosine of this angle: denoting by (A^TB) the scalar product of vectors *A* and *B*, one has

$$\cos^2(\Phi_k^{\mathrm{T}} Y_P) = \frac{(\Phi_k^{\mathrm{T}} Y_p)^2}{(\Phi_k^{\mathrm{T}} \Phi_k)(Y_p^{\mathrm{T}} Y_{\mathrm{p}})}$$

Thus, in the first step of the procedure, all *K* such cosines are computed, and the vector Φ_k giving the largest cosine is ranked first; we denote it by $\Phi_k{}^1$.

Before proceeding to rank the second vector, one has to eliminate the part of the output which is explained by $\Phi_k{}^1$. This is performed by projecting all Φ_k's, and the output, onto the subspace (of dimension N_T-1) which is orthogonal to $\Phi_k{}^1$. Then the second vector is found by the same procedure in the subspace and so on until all Φ_k's are ranked. This is illustrated in Figure 2.

Once all vectors are ranked, a selection among them must be made on the basis of the resulting ranked list. There exist rigorous hypothesis testing procedures,[16] whose application to the prediction of logP are currently under investigation in our group. In the present work, the Φ_k's were monomials, selected as follows: at each step of the Gram−Schmidt orthogonalization, in addition to ranking a monomial, the value of the parameter of the model involving the new ranked monomial was computed (at a very small computational cost); the performance of the model was estimated on a separate data set, and ranking was stopped when overfitting was detected.

The final step consisted in choosing the descriptors that appeared most frequently in the selected monomials (e.g., in monomial $x_1x_3{}^2$ descriptor x_1 is considered to appear once and descriptor x_3 is considered to appear twice).

5. PREDICTION OF LOGP

The prediction of logP from molecular properties with neural networks is still a challenging problem. While several studies based on classical regression techniques have already been performed with encouraging results,[17−20] using small data bases, further work using a neural network approach did not bring any substantial improvement upon the previous results.[21] However, neural nets should be the method of choice to address this problem, since most of the selected descriptors relevant to the prediction of logP are involved as their second or fourth power,[18,22] which indicates that the models are nonlinear. We show in the following that the failures to improve the results with neural networks can be traced to the fact that an appropriate methodology was not used (sloppy training algorithms, overparameterized networks, no input selection, etc). We further show that, if neural nets are used appropriately, the improvements predicted by theory can indeed be achieved.

The 323 molecule data base published by Bodor[19] was first chosen to apply our methodology, for the following reasons:

•it covers a reasonable range of organic functions,

•a set of descriptors has already been selected for this particular data base,

•it has a significant size with respect to the number of descriptors,

•the experimental logP data could be checked from others sources,

•it provides a basis for comparison.

5.1. Data Base Generation. All the molecules were sketched and optimized using the Biosym package[23] on an IBM RISC 6K workstation. For each molecule, the lowest energy conformation obtained using the esff force field was first selected. The Ampac AM1 method[24] version 2.1 was then used to compute the final geometry and the atomic/molecular parameters. Special care was exercised for drawing the "true" structures, in particular for molecules having internal H-bonding. For complex cases, like tetracycline which has several sites for internal H-bonding, experimental coordinates were used as initial positions.[25]

The network computations as well as the descriptor selections with the Gram−Schmidt orthogonalization technique were done with home made programs written in C.

5.2. Improvements Using Previous Descriptors. 5.2.1. Results Obtained with Neural Networks. The set of descriptors initially published by Bodor[19] was first reduced to its ten first degree descriptors (see Appendix). Neural networks with one hidden layer, designed with the resultant descriptor set as inputs and with a variable number of hidden neurons, were tested for logP prediction using both the cross validation and the leave-one-out techniques described above. The corresponding results are reported in Table 1, lines 1 and 2, together with those obtained for two smaller descriptor sets after withdrawal of some descriptors which, according to other sources,[4,26,27] were not essential. For comparison, results from a linear regression (0n17d [Training a network with zero hidden neurons and a linear output neuron is identical to performing a linear regression.]) with all the 17 descriptors used by Bodor are included.[21] The selected networks performed quite well with 10 descriptors and gave slightly better results than standard regression methods when two and four descriptors were removed. A significant decrease of the network performances was observed when the final six descriptors were withdrawn in turn.

Table 1. Standard Error of Predictions[a] (SEP) for the Neural Architecture *HnDd*, with *H* Hidden Neurons and *D* Descriptors

neural network[b]		7n8d	7n6d	
(removed descriptors)	6n10d[e]	(IA,DM)[f]	(IA,DM,O,qT)[g]	0n17d[h]
CV total base[c]	0.37	0.35	0.35	0.37
LoO total base[c]	0.36	0.34	0.36	0.37
CV reduced base[d]	0.28	0.26	0.27	0.34
LoO reduced base[d]	0.27	0.27	0.27	0.34
CV separate test base[c]	0.28	0.30	0.30	irrelevant
LoO separate test base[d]	0.32	0.31	0.31	irrelevant

[a] Average of three runs. [b] The number of hidden neurons *H* was optimized to give the best SEP for a given number of descriptors *D*. [c] ±0.02. [d] ±0.01. [e] No. of parameters (P): 73. [f] No. of parameters (P): 71. [g] No. of parameters (P): 57. [h] No. of parameters (P): 18.

With all the neural architectures tested, the SEP varied significantly (5%) for different partitions into training set and evaluation set: therefore, outlier molecules that could account for such a behavior, were investigated. A molecule was considered to be an outlier when the deviation between the predicted logP and the experimental value was larger than 0.8. That threshold was reached, *with all architectures used*, for seven molecules (caffeine, methadone, penicillin, 1,4-pentadiene, perylene, prostaglandin, tetracycline), which were therefore temporarily discarded from the data base. The case of these molecules is discussed in section 5.2.3.

For that reduced base, the SEP was improved as expected (by 25%) with equivalent results for all networks; moreover the results were less scattered (Table 1, lines 3 and 4). In addition, the networks performed better than Bodor's polynomial regression (0n17d).

The network ability to generalize was then tested, with a test base of 48 compounds, given in Appendix,[22] by the same methods as above. On this specific data set, all the architectures (Table 1, lines 5 and 6) provided good predictions, since a SEP value of 0.30 came quite close to the experimental logP error (0.2−0.3).[18] For comparison, polynomial regression performed on the reduced data base with Bodor's 17 descriptors gave a SEP of 0.41 on the separate test base, which is about 25% larger than the results obtained by the neural networks.

This shows clearly that using the initial 17 descriptors was unnecessary. A simple network built with only six inputs, namely nC, S, MW, qO, qN, and qNO (see Appendix), and seven hidden neurons, trained with about 250 molecules, was actually quite appropriate for the logP prediction of the studied compounds. Moreover, the above results would fit well with the widely-held view that logP depends mostly on solute size and hydrogen-bond basicity,[26] since one half of the descriptors are size-shape parameters and the other half relates to the solute hydrogen-bond acceptor strength. Thus, the electronic descriptors built from the calculated atomic charges on oxygen and nitrogen atoms would play a dominant role and could be parameters describing hydrogen bonding. A special indicator introduced for alkane, whose relevance seemed dubious to other QSAR researchers,[27] actually appears to be irrelevant; so does the computed dipole.[26]

For comparison with a regression approach proposed previously, we report in Table 2 predictions of logP performed with our best model and with Bodor's 18-parameter function;[19] for this test, a set of 27 molecules, not members of the main data set, were chosen among a set of 107 molecules whose logP were estimated by regression,[19] because a reliable experimental logP was available.[28] An overall improvement of 30% is observed.

Table 2. Comparison of the Prediction Performance of the Network Approach with a Classical Regression with 18-Parameter Function

compd		exptl	this	predicted
no.	name	logP	work[a]	Bodor
1	methane	1.09	0.77	1.38
2	ethane	1.81	1.35	1.79
3	methylcyclopentane	3.37	3.19	2.64
4	methylcyclohexane	3.61	3.7	3.09
5	cycloheptane	4	3.65	3.09
6	trans-1,2-diphenylethylene	4.81	4.89	4.88
7	dichlorodifluoromethane	2.16	1.62	1.82
8	diiodomethane	2.3	2.96	3.26
9	bromomethane	1.19	1.08	1.42
10	2-bromopropane	2.14	2.32	2.21
11	1,3,5-tribromobenzene	4.51	4.75	4.89
12	1,3-dibromobenzene	3.75	3.97	4.05
13	1,2-difluorobenzene	2.37	2.50	2.62
14	1,4-diiodobenzene	4.11	4.53	4.78
15	3-PCB	4.58	4.63	4.66
16	2-pentanol	1.19	1.27	1.4
17	2-hexanol	1.76	1.86	1.94
18	3-hexanol	1.65	1.86	1.9
19	methyl *n*-propyl ether	1.21	0.83	0.9
20	methyl *n*-butyl ether	1.66	1.40	1.44
21	methyl *tert*-butyl ether	0.94	1.34	1.36
22	3-methyl-2-butanone	0.84	1.11	1.08
23	4-methyl-2-pentanone	1.31	1.71	1.6
24	2,4-dimethyl-3-pentanone	1.86	2.29	2.1
25	methyl propionate	0.82	0.71	0.64
26	methyl butyrate	1.29	1.21	1.2
27	isobutyl acetate	1.78	1.72	1.74
		SEP	0.30	0.40

[a] Average of cross-validation with 100 cycles (7n6d).

5.2.2. Numerical Demonstration of the Parsimony of Neural Nets. As mentioned in section 2, the sole justification of using neural nets in QSAR, in lieu of standard nonlinear regression techniques such as polynomial regression, is the *parsimony* of the former. This has been proven mathematically;[3] it is demonstrated numerically in this section.

Polynomial regressions were performed with the 10 descriptors and the six descriptors mentioned in the previous section, using the complete data base.

·*With 10 descriptors*, a regression of degree 2 has 66 parameters (1 constant term, 10 linear terms, and 55 second degree terms). This is roughly equivalent to the number of parameters present in the smallest network (six descriptors and seven hidden neurons, hence 57 parameters) which gives rise to a SEP of 0.35 (Table 1) on the test set. With this polynomial regression, the SEP (averaged over 100 partitions into training set and validation set) on the training set was 0.24 and the average SEP on the validation set was 0.45, thereby showing extensive overfitting, as expected.

It might be argued that the difference arises from the fact that the polynomial regression has more parameters than the neural regression; this is not the case: if one selects, with the Gram−Schmidt procedure, the first 57 monomials—thus providing a polynomial regression with exactly the same number of parameters as the neural network with six inputs and seven hidden neurons—the training and validation SEP are 0.24 and 0.44, respectively, which shows that overfitting is still present.

It might be further argued that polynomials might provide better results if a smaller number of monomials were used; such is not the case. With polynomial regression, the best result (SEP of 0.33 on the training set and 0.41 on the validation set) was obtained with the first 16 monomials ranked by the Gram−Schmidt procedure. This result is still inferior by 15% to the result obtained by neural regression.

·*With six descriptors*, polynomial regression of degree 2, was similarly performed. It resulted in a 27-parameter model with a training SEP and validation SEP of 0.32 and 0.45, respectively. Monomial ranking and selection resulted in a 23-parameter model with a training SEP of 0.33 and a validation SEP of 0.43. Again, the SEPs are significantly larger than those obtained by neural networks.

Roughly similar results were obtained with the reduced data base.

The results presented in this section demonstrate clearly the advantage of neural networks over polynomials for nonlinear regression: (i) for a given number of parameters, neural networks, when trained with efficient algorithms and provided with relevant descriptors, yield better predictions; (ii) even when monomial selection is used for polynomial approximation, neural networks outperform polynomials. This is a mathematically proven property, so that the fact that it applies to QSAR (as well as to any other field where nonlinear regression is needed) is no surprise.

5.2.3. Discussion of the Seven Outliers. The seven molecules previously set apart were examined to detect possible errors. A careful checking of their experimental logP values in other reliable data bases[28−30] showed that 4 logP data (methadone, perylene, prostaglandin, 1,4-pentadiene) used by Bodor were not correct. It turned out that our 7n6d network predictions for these compounds were actually correct as illustrated by the predicted and corrected experimental logP for these compounds: methadone (predicted: 4.1, corrected: 3.93), perylene (5.7, 5.82), prostaglandin (2.8, 2.0), and 1,4-pentadiene (2.4, 2.48). This is a good illustration of the ability of a neural network to detect errors or inconsistencies in the experimental data used to feed the model.

As for tetracycline, which had the largest logP deviation, it was incorrectly drawn since at neutral pH it exists as a zwitterion.[31,32] Its predicted logP did not take into account that contribution and therefore was given with a large excess. A similar inaccuracy in logP prediction was observed for the amino acids also predominant in the zwitterionic form at pH 7 (results not shown). All attempts to compute this particular charge effect failed to improve the prediction, presumably because some extra descriptor(s) might be needed. However, throwing in new descriptors is pointless as long as new molecules, exhibiting similar effects, are not included in the data base.

Finally, we considered that the prediction of the two remaining outliers, penicillin and caffeine, could probably be improved with the introduction of new computed descriptors, which was the next step of our investigation (described in section 5.3.2 below). Therefore, these molecules were kept in the data base for further investigations.

The above results, as mentioned before, stressed the importance of the electronic descriptors, and it appeared likely that new descriptors, containing atomic and group charge information, would help in correlating the special electronic effects of these specific compounds with their partition coefficient. It turned out that this was indeed the case, as shown in section 5.3.2 below.

Table 3. Standard Error of Prediction (SEP) and Maximum logP Deviation for the *H*n*D*d Networks for the 321 Molecule Set

neural network	SEP	logP deviatn
7n6d	0.30	1.6
6n7d	0.28	1.3
5n18d[a]	0.261[b]	0.8

[a] Average of three runs. [b] ±0.002.

In view of the above results, the experimental logP values for the complete data base were checked.[28] In addition to the above-mentioned four molecules, large discrepancies were found between experimental logP values of hexachlorophene published in various sources; therefore, this molecule was removed permanently from the data base. Tetracycline was also discarded because of its zwitterionic character in water.

To summarize: as a consequence of the discussion reported in this paragraph, the results reported in the rest of the paper were obtained with a set of 321 molecules, which differs from the data base used to obtain the results shown under the heading “reduced set” in Table 1 by the facts that (i) four molecules (methadone, perylene, prostaglandin, 1,4-pentadiene) were reintroduced after correction of the experimental values of their logP, (ii) penicillin and caffeine were reintroduced for further investigation, (iii) tetracycline was permanently discarded because of its zwitterionic character, and (iv) hexachlorophene was permanently discarded because of the uncertainty on its logP value.

5.3. Automatic Descriptor Selection. 5.3.1. Selection from the Ten Previous Candidate Descriptors. The descriptor selection method described in section 4 was first checked on this new data set with the 10 descriptors used in the previous section. The first seven ranked descriptors (using the Gram−Schmidt method with monomials up to degree 3) were the six finally selected above, plus qT. It can be seen from Table 3 that a network built with those seven inputs yielded the best performances, although some compounds, like caffeine, were still predicted with a deviation greater than or close to unity. This method gave a good agreement with the “heuristic” selection described in section 5.2 and furthermore led to the design of a more efficient network with six hidden neurons and seven descriptors.

5.3.2. Selection from an Extended Pool of Candidate Descriptors. As discussed in section 5.2.3, it was deemed necessary to introduce new candidate descriptors in order to improve the accuracy of the prediction for some molecules. Therefore, a new set of 74 descriptors, summarized in the Appendix, was designed. Out of these, 18 relevant descriptors were selected with the Gram−Schmidt orthogonalization technique (with monomials up to degree 2). The results of three leave-one-out cycles with a network built with those descriptors are reported in Table 3. Although the *overall* improvement is slight, it is due essentially, as expected, to a large improvement in the prediction of (i) the previous outliers (penicillin and caffeine) and (ii) the polyfunctional compounds (for which the SEP dropped from 0.40 to 0.33). Thus, our selection method allows us to try whatever descriptors one may think of and quickly assess their relevance and ability to enlarge the range of compounds whose logP can be accurately predicted.

Further studies, making use of our new selection procedure, will be necessary to assess the relevance of the new selected descriptors on larger data bases. Such investigations, along with prediction of other physico-chemical properties (boiling point and water solubility) using the presented methodology are currently in progress in our group.

6. CONCLUSION

We have reported an investigation of the use of neural networks for the statistical prediction of logP. In the first part of the paper, we described elements of a principled approach to the use of neural networks in data modeling; we explained why neural networks may give superior results as compared to those of other regression techniques and how they should be used in order to actually get such results in a statistically reliable way. In addition, we introduced an original procedure for descriptor selection. The application of these concepts and methods to the prediction of logP was reported in the last part of the paper; first, the parsimony of neural networks was demonstrated numerically; it was subsequently shown that the automatic descriptor selection technique gives results similar to those obtained by combining chemical insight with a trial-and-error approach. Improvements in the accuracy obtained by using a selection among a larger pool of candidate descriptors was also reported. Our future work is oriented toward (i) increasing the size of the database and finding new relevant descriptors and (ii) automating the choice of the number of hidden neurons based on a rigorous statistical method.

APPENDIX

The 10 first order descriptors used by Bodor are reported below:

MW	molecular weight
S	molecular surface
O	ovality of the molecule
nC	number of carbon atoms
IA	Boolean indicator for alkanes
qO	square root of the sum of the squared charges on oxygen atoms
qN	square root of the sum of the squared charges on nitrogen atoms
qNO	sum of the absolute values of atomic charges on nitrogen and oxygen atoms
qT	sum of the absolute values of atomic charges on each atom
DM	computed dipole moment

For the atom Z equal to C, N, O, F, Cl, Br, I, S, the following 74 descriptors were computed:

nZ	number of Z atoms
AqZ	sum of the absolute values of atomic charges on Z atoms
qZ	sum of the atomic charges on Z atoms
q2Z	sum of the squared atomic charges on Z atoms
AqmZ, qmZ, q2mZ	average of the corresponding descriptors, i.e., divided by nZ
S, O, DM, MW, qNO, qT	same as above
qTm	qT divided by the molecule atom number
qNOm	sum of AqmO and AqmN
qH	sum of the atomic charges on labile hydrogen atoms
nCar	number of aromatic carbon atoms
nCak	number of carbon atoms simply bonded either to hydrogen or carbon atoms
qCal	sum of the atomic charges on carbon α to oxygen in alcohols and ethers
qOal	sum of the atomic charges on ether and alcoholic oxygen atoms
qCac	sum of the atomic charges on carbon atoms in carboxylic acid and ester groups
qCO2	sum of the atomic charges on the three atoms of the carboxylic acid and ester groups
qCest	sum of the atomic charges on the α oxygen bonded carbon atom of carboxylic ester groups
qCph	sum of the atomic charges on oxygen atoms in phenols
qNam	sum of the atomic charges on nitrogen atoms in amines

nCar, nCak, nF, nCl, nBr, nO, nN, O, MW, D, qC, qmC, qmN, aqmO, qNOm, qOAl, qCac, and qNam were used as inputs in the 5n18d network.

The 48 molecule test set is as follows: methane, ethane, cycloheptane, methylcyclopentane, methylcyclohexane, 2-pentanol, 2-hexanol, 3-hexanol, 3-methyl-2-butanone, 4-methyl-2-pentanone, pinacolone, diipropylketone, amylamine, ethylamine, hexylamine, heptylamine, butyl methyl ether, methyl propyl ether, methyl *tert*-butyl ether, methylbutyrate, propionic acid methyl ester, iso-butylacetate, 2-nitropropane, methylbromide, 2-bromopropane, 2-iodopropane, dibromomethane, diiodomethane, 1,2-dibromoethane, 1,1,2-trichloroethane, tribromomethane, 1,1,2,2-tetrachloroethane, *cis*-1,2-dichloroethylene, *trans*-1,2 dichloroethylene, dichlorodifluoromethane, pentachloroethane, hexachloroethane, 1,1,2 trichlorotrifluoroethane, freon114, *trans*-stilbene, butyronitrile, prednisolone, 3-chlorobiphenyl, 1,2-difluorobenzene, 1,3-dibromobenzene, 1,4-diiodobenzene, 1,3,5-tribromobenzene, heptan-4-ol.

REFERENCES AND NOTES

(1) Bishop, C. *Neural networks for pattern recognition*; Oxford University Press: New York, 1995.

(2) Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Networks* 1989, *5*, 359−366.

(3) Hornik, K.; Stinchcombe, M.; White, H.; Auer, P. Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Comp.* **1994**, *6*, 1262−1275.

(4) Cense, J. M.; Diawara, B.; Legendre, J. J.; Roullet, G. Neural networks prediction of partition coefficients. *Chem. Intel. Lab. Syst.* **1994**, *23*, 301−308.

(5) Grunenberg, J.; Herges, R. Prediction of chromatographic retention values (R_M) and partition coefficients (log P_{oct}) using a combination of semiempirical self-consistent reaction field calculations and neural networks. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 905−911.

(6) Press, W. H.; Teukolski, W. T.; Vetterling, W. T.; Flannery, B. P. *Numerical receipe in C: the art of scientific computing*; Cambridge University Press: Cambridge, 1992.

(7) Wessel, M. D.; Jurs, P. C. Prediction of reduced ion mobility constants from structural information using multiple linear regression analysis and computational neural networks. *Anal. Chem.* **1994**, *66*, 2480−2487.

(8) Breindl, A.; Beck, B.; Clark, T.; Glen, R. C. Prediction of the n-octanol/water partition coefficient, logP, using a combination of semiempirical MO-calculations and a neural network. *J. Mol. Model.* **1997**, *3*, 142−155.

(9) Nefati, H.; Cense, J. M.; Legendre, J. J. Prediction of the impact sensitivity by neural network. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 804−810.

(10) Stoppiglia, H. Méthodes Statistiques de Sélection de Modèles Neuronaux; Applications Financières et Bancaires. Ph.D. Theses. Université Pierre et Marie Curie: Paris, 1997.

(11) Egolf, L. M.; Wessel, M. D.; Jurs, P. C. Prediction of boiling points and critical temperatures of industrially important organic compounds from molecular structure. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 947−956.

(12) Mitchell, B. E.; Jurs, P. C. Prediction of autoignition temperatures of organic compounds from molecular structure. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 538−547.

(13) Tetko, I. V.; Villa, A. E. P.; Livingstone, D. J. Neural network studies. 2. Variable selection. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 794−803.

(14) Sutter, J. M.; Dixon, S. L.; Jurs, P. C. Automated descriptor selection for quantitative structure-activity relationships using generalized simulated annealing. *J. Chem. Inf. Comp. Sci.* **1995**, *35*, 77−84.

(15) Chen, S.; Billings, S. A.; Luo, W. Orthogonal least square methods and their application to non-linear system identification. *Int. J. Cont.* **1989**, *50*, 1873−1896.

(16) Leontaritis, I. J.; Billings, S. A. Model selection and validation for nonlinear systems. *Int. J. Cont.* **1987**, *45*, 311−341.

(17) Klopman, G.; Iroff, L. D. Calculation of partition coefficients by the charge density method. *J. Comput. Chem.* **1981**, *2*, 157−160.

(18) Bodor, N.; Gabanyi, Z.; Wong, C.-K. A new method for the estimation of partition coefficient. *J. Am. Chem. Soc.* **1989**, *111*, 3783−3786.

(19) Bodor, N.; Huang, M.-J. An extended version of a novel method for the estimation of partition coefficients. *J. Pharm. Sci.* **1992**, *81*, 272−281.

(20) Klopman, G.; Namboodiri, K.; Schochet, M. Simple method of computing the partition coefficient. *J. Comput. Chem.* **1985**, *6*, 28−38.

(21) Bodor, N.; Huang, M.-J.; Harget, A. Neural network studies. 3. Prediction of partition coefficients. *J. Mol. Struct. (Theochem.)* **1994**, *309*, 259−266.

(22) Bodor, N.; Huang, M.-J. A new method for the estimation of the aqueous solubility of organic compouds. *J. Pharm. Sci.* **1992**, *89*, 954−960.

(23) Insight/Discover, Release 95.0; Biosym/MSI: San Diego, CA, 1995.

(24) Dewar, J. S.; Zoebisch, E. G.; Healy, E. F.; Stewart, J. J. P. AM1: a new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc.* **1985**, *107*, 3902−3909.

(25) Cambridge Crystallographic Database, Version 5.12; Cambridge Crystallographic Data Center: 12 Union Road, Cambridge, CB2 1EZ, U.K, 1995.

(26) Leo, A. Calculating log Poct from structures. *Chem. Rev.* **1993**, *93*, 1281−1306.

(27) Hansch, C.; Leo, A. *Exploring QSAR, Fundamentals and applications in chemistry and biology*; American Chemical Society: Washington, DC, 1995; Vol. 1.

(28) Hansch, C.; Leo, A.; Hoekman, D. *Exploring QSAR, Hydrophobic, Electronic, and steric Constants*; American Chemical Society: Washington, DC, 1995; Vol. 2.

(29) Klopman, G.; Li, J.-Y.; Wang, S.; Dimayuga, M. Computer automated logP calculations based on an extended group contribution approach. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 752−781.

(30) CMC3D: MDL Information Systems Inc.: San Leandro, CA, 1996.

(31) Stezowski, J. J. Chemical structure properties of tetracycline derivatives. Molecular structure and conformation of the free base derivatives. *J. Am. Chem. Soc.* **1976**, *98*, 6012−6018.

(32) Pliska, V.; Testa, B.; van de Waterbeemd, H. *Lipophilicity in drug action and toxicology*; VCH: Weinheim, 1996; Vol. 4.

CI980042V