ELSEVIER

Journal of Chromatography A, 904 (2000) 119–129

JOURNAL OF CHROMATOGRAPHY A

www.elsevier.com/locate/chroma

Artificial neural networks in liquid chromatography: efficient and improved quantitative structure–retention relationship models

Yannis L. Loukas*

Department of Pharmaceutical Chemistry, School of Pharmacy, University of Athens, Panepistimiopolis, Zografou, 157 71 Athens, Greece

Received 26 April 2000; received in revised form 8 September 2000; accepted 8 September 2000

Abstract

The application of the principal neural network architecture, namely the multilayer perceptron (MLP), has been developed for obtaining sufficient quantitative structure–retention relationships (QSRR) with high accuracy. The present study is an extension to the excellent study of Cserhati et al. [LC–GC Int., 11 (1998) 240] for the retention behavior of solutes based on their structure. To this end, a dataset of 25 substances as solutes to two different stationary phases (polyethylene–silica and polyethylene–alumina) were analyzed to their structural descriptors and related to their retention behavior as expressed by the logarithms of their capacity factors (log k'). The results were compared to those of Cserhati et al. who studied the same problem using as many as ten different equations based on multiple regression analysis. In the present study a series of new and improved algorithms other than the 'old-fashioned' and problematic steepest descent were examined for training the MLP networks. The proposed methods led to substantial gain in both the prediction ability and the computation speed of the resulting models. For the development and evaluation of the artificial neural network (ANN) systems the same (eight) descriptors proposed by Cserhati were used also in this study. Furthermore, the results were compared to those produced from classical linear multivariate regression such as partial least squares regression (PLS). Some of the proposed ANN models diminished the number of outliers, during their implementation to unseen data (solutes), to zero.

Keywords: Artificial neural networks; Quantitative structure–retention relationships

1. Introduction

The interest in studies dealing with the retention behavior of solutes in different stationary phases has been increased substantially the last years [1]. In these studies the major aim is to develop improved expert systems capable to predict and describe the retention capability of different stationary phases, for the better understanding of the retention process, and to provide a valuable chromatographic tool for highlighting the molecular mechanisms of retention in a given HPLC system. In almost all of these studies much effort is concentrated on the calculation of structural descriptors, which characterize the examined solutes. In the literature several models have been described, from linear to nonlinear ones in order to calculate these values as accurately as possible [2]. In the present study the application of ANN is being

*Tel.: +30-1-7274-224; fax: +30-1-6130-285.

E-mail address: loukas@pharm.uoa.gr (Y.L. Loukas).

0021-9673/00/$ – see front matter
PII: S0021-9673(00)00923-7

described, based on advanced algorithms, in order to predict accurately the log k' values of 25 structurally diverse aromatic solutes (Table 1) in two stationary phases in isocratic HPLC systems and to provide efficient QSRR models.

QSRR studies can be used for the selection of principal physicochemical characteristics (descriptors), their relation to retention values and the derivation of mathematical models that involve these multivariate data in order to be used for predictive purposes in every HPLC system. Multivariate data consist of the results of observations of many different variables (physicochemical descriptors) for a number of individuals (molecules). Each variable may be regarded as constituting a different dimension, such that if there are n variables each object may be said to reside at a unique position in an abstract entity referred to as n-dimensional hyperspace. This hyperspace is necessarily difficult to visualize, and the underlying theme of multivariate analysis (MVA) is thus the description of a polynomial in which the dependent variables are related to the independent variable(s). Known methods for this include the multiple regression analysis, experimental design techniques, nonlinear regression. The drawback, sometimes, of these very popular techniques is their inability to give highly predictive models due to hidden nonlinearity inside the data variables or the prerequisite to specify the mathematical model before the fitting of the data. So there is a need to improve further such kind of models in order to extract the most accurate prediction. To this end artificial neural networks (ANN) [3] and especially the 'supervised' ones could be used successfully in QSRR studies providing better results than the conventional regression models.

In the following study we shall examine the behavior of a series of training algorithms in the

Table 1
Structures, physicochemical parameters and observed log k' values of the examined solutes in two stationary phases[a]

	R_2	p_2^H	Σa_2^H	$\Sigma \beta_2^H$	V_x	μ	d_{max}	V_{aq}	log $k'(1)$	log $k'(2)$
Hexylbenzene	0.591	0.5	0	0.15	1.562	0.351	0.1326	649.7	4.56	3.721
1,3,5-Tris(1-methylethyl)benzene	0.627	0.4	0	0.22	1.985	0.014	0.1309	777.15	4.887	4.147
1,4-Dinitrobenzene	1.13	1.63	0	0.41	1.065	0	0.5652	451.02	0.969	0.774
3-(Trifluoromethyl)phenol	0.425	0.87	0.72	0.09	0.969	2.096	0.4649	432.38	0.975	0.995
3,5-Dichlorophenol	1.02	1.1	0.83	0	1.02	1.408	0.2239	440.2	1.502	1.528
4-Hydroxybenzonitrile	0.94	1.63	0.79	0.29	0.93	3.313	0.2237	409.94	0.396	0.372
4-Iodophenol	1.38	1.22	0.68	0.2	1.033	1.586	0.2213	434.24	1.174	1.173
Methoxybenzene	0.708	0.75	0	0.29	0.916	1.249	0.1481	407.98	0.835	0.589
Benzamide	0.99	1.5	0.49	0.67	0.973	3.583	0.3448	418.21	0.303	−0.069
Benzene	0.61	0.52	0	0.14	0.716	0	0.1301	331.81	0.584	0.313
Chlorobenzene	0.718	0.65	0	0.07	0.839	1.307	0.1466	375.23	1.129	0.916
Cyclohexanone	0.403	0.86	0	0.56	0.861	2.972	0.1111	383.84	0.337	0.867
Dibenzothiophene	1.959	1.31	0	0.18	1.379	0.524	0.4465	555.67	3.041	3.126
Phenol	0.805	0.89	0.6	0.3	0.775	1.233	0.2173	353.02	0.099	0.047
1,1,2,3,4,4-Hexachloro-1,3-butadiene	1.019	0.85	0	0	1.321	0.001	0.0606	516.73	3.248	3.426
1H-Indazole	1.18	1.25	0.54	0.34	0.905	1.546	0.2752	405	0.822	0.647
3,7-Dihydro-1,3,7-trimethyl-1-H-purine-2,6-dione	1.5	1.6	0	1.35	1.363	3.708	0.401	569.32	1.616	1.042
4-Nitrobenzoic acid	0.99	1.07	0.62	0.54	1.106	3.431	0.5643	467.67	−0.899	−0.924
1-Methyl-2-pyrrolidone	0.491	1.5	0	0.95	0.82	3.594	0.307	381.5	0.257	−0.699
Naphthalene	1.34	0.92	0	0.2	1.085	0	0.132	458.91	1.769	1.583
4-Chlorophenol	0.915	1.08	0.67	0.2	0.898	1.478	0.2201	396.25	0.758	0.758
Methylbenzene	0.601	0.52	0	0.14	0.716	0.263	0.1301	384.44	1.027	0.829
Piperazine	0.57	0.83	0.2	1.17	0.763	1.995	0.1583	355.56	0.797	0.252
Piperidine	0.422	0.46	0.1	0.69	0.804	1.168	0.1554	368.5	0.574	−0.021
Benzonitrile	0.742	1.11	0	0.33	0.871	3.335	0.1451	388.93	0.705	0.337

[a] R_2, excess molar refraction; p_2^H, solute dipolarity/polarizability; Σa_2^H, solute overall hydrogen bond acidity; $\Sigma \beta_2^H$, solute overall hydrogen bond basicity; V_x, McGowan characteristic volume; μ, total dipole moment; d_{max}, electron excess charge on an atom in solute molecule; V_{aq}, solvent (water) accessible molecular volume.

behavior of feedforward neural networks and the results will be compared with published ones [4] as well as with the results from the PLS method [5].

2. Method

2.1. ANN topology

ANN topologies [6–9], or architectures, are formed by organizing nodes into layers and linking these layers of neurons with modifiable weighted interconnections. A diagrammatic representation of the ANN used in the present study consisting of 8 inputs (the proposed descriptors) and 1 output (log k' values) connected to each other by 1 hidden layer consisting of 5 nodes (5 was chosen through trial and error) is shown in Fig. 1. In the fully connected topology shown the 8 nodes in the input layer are connected to the 5 nodes in the hidden layer, by 40 connection weights, which in turn are connected to the 1 output node, by a further 5 connection weights. In addition, there is also a bias (extra node), which always has an activation level of +1, which is connected to nodes in the hidden and output layers (but not the input layer) via modifiable weighted connections (extra 6 connections). Such architecture can be written as 8-5-1 ANN and is commonly referred to as a fully interconnected feedforward multilayer perceptron. The examined architecture 8-5-1 generates 51 adjustable parameters when there are only 25 observations and someone could ask for possible overfitting during ANN training. In ANN applications nets are often used with many times more parameters than training cases. Nelson and Illingworth [10] discuss training a network with 16 219 parameters with only 50 training cases. Furthermore, Lawrence et al. [11] have shown that it can be difficult to reduce training error to a level near globally optimal value, even when using more weights than training cases. But increasing the number of weights makes it easier to find a good local optimum, so using 'oversize' networks can reduce both training and testing error. Sarle [12] suggests that if it is used early stopping (as in the present case — see later) it is essential to use lots of hidden units to avoid bad local optima. Finally Weigend [13] suggests that, if it used early stopping, there seems to be no upper limit on the number of hidden units other than that imposed by computer time and memory requirements.

In the present study in order to avoid overfitting early stopping proceeded as follows: (a) division of the available data into training and validation sets (b)

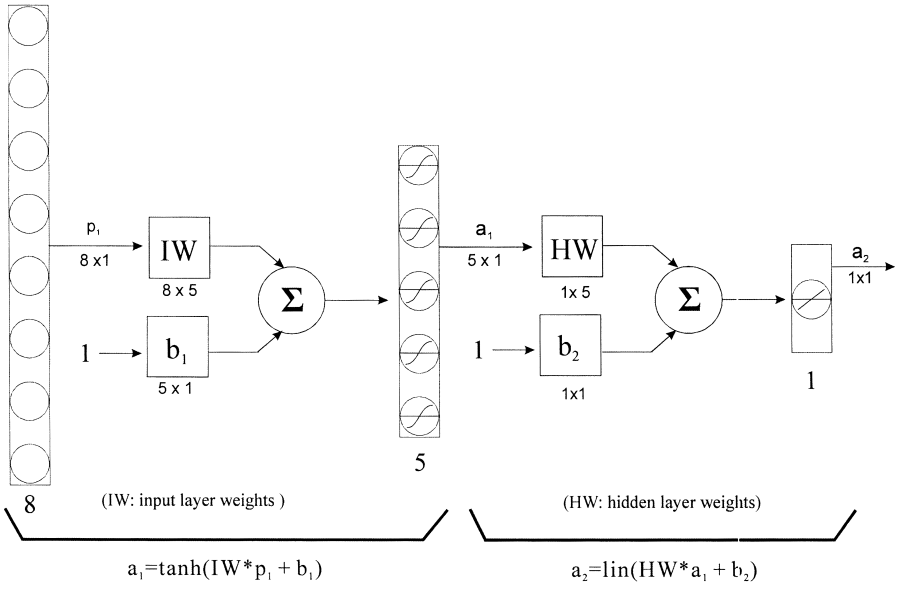

Fig. 1. Schematic representation of the architecture and the way of processing of the examined MLP network.

usage of a large number of hidden units and (c) compute the validation error during training (d) stop training when the validation error rate start to deteriorate. Early stopping has several advantages: it is fast, it can be applied successfully to networks in which the number of weights far exceeds the sample size (as in the present study) and it requires only the division of the data set into training and validation (see later).

The above 8-5-1 architecture is rather simple and the question arises ''How does one choose the number x of nodes in the hidden layer?'' There is no theory yet to suggest how many hidden unit are needed to approximate any given function. Some rule of thumb relates the total number of trainable weights in the network to the number of training cases. A typical recommendation is that the number of weights should be no more than 1/30 of the number of training cases. Such do not apply when regularization (such as early stopping) is used [12]. For the present study the 'trial and error' was adopted starting from 10 hidden units and reducing their number to 1 unit. The network with 5 units gave the best generalization.

2.2. Training the neural network

The first step is to choose the input variables and the dataset. In the present study we use the dataset of 25 aromatic compounds (Table 1) acting as solutes to an isocratic HPLC system with two different stationary phases and as inputs eight physicochemical descriptors (Table 1). Before training a feedforward network, the weights and biases must be initialised. For feedforward networks with sigmoid layers, as in the present study, the function of choice for initialisation is based on the technique of Nguyen and Widrow [14]. After that the inputs and targets are normalized so that they will have zero mean and unity standard deviation.

Once the network weights and biases have been initialised, and the scaling has been applied, the network is ready for training. The network can be trained for function approximation (nonlinear regression). The training process requires a set of examples of proper network behavior — network inputs and target outputs. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function. The default performance function for feedforward networks is the mean square error (MSE) — the average squared error between the network outputs and the target outputs.

In the beginning of the training two backpropagation training algorithms were used: gradient descent and gradient descent with momentum. These two methods are often too slow for practical problems. To overcome this problem we examined several high-performance algorithms, which can converge from 10 to 100 times faster than the algorithms examined previously. All of these algorithms train faster and fall into two main categories. The first category uses heuristic techniques, which were developed from an analysis of the performance of the standard steepest descent algorithm. Such heuristic modifications include the use of momentum, variable learning rate backpropagation, and resilient backpropagation. The second category of fast algorithms uses standard numerical optimization techniques such conjugate gradient, quasi-Newton and Levenberg–Marquardt (Table 2).

In most of the training algorithms that we have already examined, a learning rate is used to determine the length of the weight update (step size). In most of the conjugate gradient algorithms the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size which will minimize the performance function along that line. There are different search functions that best suited to certain training functions, although the optimum choice can vary according to the specific application and could be found with trial and error. In the present study, for the conjugate gradient algorithms we used the Charalambous' search algorithm [15], since it appeared to produce excellent results while for the quasi-Newton algorithms we got best results with the backtracking algorithm [16].

Newton's method is an alternative to the conjugate gradient methods for fast optimization. It requires the calculation of the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton's method often converges faster than conjugate gradient methods. Unfortunately, it is complex and expensive to compute the Hessian matrix for feedforward neural

Table 2
Algorithms used for training the standard feed-forward MLP networks and the resulted mean square error, post training correlation coefficient R and the number of outliers

Algorithm	MSE	Post train R	Outliers[a]
Levenberg–Marquardt backpropagation	0.018 (0.023)[b]	0.996 (0.992)	0 (0)
Polak-Ribiere conjugate gradient backpropagation	0.025 (0.031)	0.989 (0.99)	1 (0)
BFGS quasi-Newton backpropagation	0.031 (0.038)	0.971 (0.972)	1 (2)
Scaled conjugate gradient backpropagation	0.041 (0.046)	0.965 (0.969)	2 (1)
Powell-Beale conjugate gradient backpropagation	0.042 (0.045)	0.955 (0.957)	3 (3)
Fletcher-Powell conjugate gradient backpropagation	0.041 (0.039)	0.956 (0.962)	3 (2)
Gradient descent w/momentum and adaptive lr backprop.	0.075 (0.079)	0.855 (0.859)	4 (3)
Gradient descent w/adaptive lr backpropagation	0.12 (0.093)	0.819 (0.850)	4 (3)
One step secant backpropagation	0.067 (0.065)	0.839 (0.822)	4 (4)
Resilient backpropagation (Rprop)	0.063 (0.075)	0.855 (0.802)	4 (4)
Gradient descent backpropagation	0.19 (0.23)	0.62 (0.73)	4 (5)
Gradient descent w/momentum backpropagation	0.278 (0.293)	0.65 (0.65)	5(5)

[a] An outlier had $|\log k'_{obs} - \log k'_{pred}| \geq 0.2$.
[b] Numbers in parenthesis correspond to values with the second stationary phase.

networks. There is a class of algorithms that are based on Newton's method but which do not require calculation of second derivatives. These are called quasi-Newton (or secant) methods. The quasi-Newton method which has been used most successful in published studies is the Broyden, Fletcher, Goldfarb and Shanno (BFGS) [17].

Like the quasi-Newton methods, the Levenberg–Marquardt (LM) algorithm [18,19] was designed to approach second-order training speed without having to compute the Hessian matrix. Instead it calculates the Jacobian matrix, which contains first derivatives of the network errors with respect to the weights and biases. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix. LM is an advanced non-linear optimization algorithm that can be used to train the weights in a network, just as back propagation would be. It is usually the fastest and most reliable algorithm available for such training. The LM algorithm has established its reputation from nonlinear regression problems were it is used successfully, as in the case of capacity factor calculations in enantiomeric separations [20] or on binding constant calculations in complexation of cyclodextrins with fluorescent compounds [21].

All these algorithms can, however, prove particularly useful for problems where high precision is required as in the case of QSRR predictions. In the present study a series of algorithms (Table 2) recently applied to MLP networks and their efficacy (in terms of computational time, number of epochs, minimized error criterion) will be compared with that of the traditional ones (steepest descent with momentum and learning rate). As will be shown later the performances differ significantly making the algorithm selection an important step in the design of ANNs.

2.3. Linear multivariate regression

The predictive ability of the examined ANN was compared further to that of classical multivariate regression. A popular technique for multivariate regression is the partial least squares (PLS) regression [5] with cross-validation as an important concept that helps to identify the appropriate number of factors (or latent variables, lv) to use. Generally, PLS can be used to develop regression models that relate a number of independent or predictor variables (X-block) to one or more dependent or predicted variables (Y-block). PLS relies on a decomposition of the X-block (the eight descriptors in the present study) based on covariance criteria. PLS finds factors (latent variables) that are descriptive of X-block variance and are correlated with the Y-block ($\log k'$). PLS is advantageous to ordinary multiple linear regression (MLR) since it examines for collinearities in the

predictor variables (i.e. some variables are linear combinations of other variables). The PLS model converges to MLR solution if all latent variables are included in the model. There are several ways to calculate PLS models, with the most commonly used the non-iterative partial least squares (NIPALS) and the SIMPLS algorithms, both of them giving exactly the same results for univariate *y*. The computational approaches for these two algorithms are well described in textbooks and are beyond the scope of the present study.

3. Results

The algorithms and the architectures described briefly in the previous section highlight almost all the available possibilities in designing effectively MLP networks with high accuracy. We have performed a QSRR study using the data in Table 1. The correlation matrix (not shown) of the examined inputs highlighted a strong linear relationship (collinearity) between inputs 5 and 8 and one should decide to exclude one of these variables from the input layer. Since the purpose of the present study is the comparison to the published results (where inputs 5 and 8 are present) it was decided not to exclude one of these inputs from the architecture of ANN and PLS models. This particular set of solutes has been studied already and is ideal for the purpose of comparison. The neural network systems were simulated using MATLAB NEURAL NETWORK TOOLBOX [22] running on a Pentium II platform. The input data were scaled before training. Training continued until there was no further decrease in overall error after a period of 500 cycles and the average training time for each run was few minutes for the examined feedforward networks. Three-layer neural networks, with eight input units and one output unit, were simulated in all cases. The eight inputs correspond to the eight descriptors and the one output to the log k' values (Table 1). The quality of QSRR was assessed by three statistical variables: (MSE, post-training least squared correlation coefficient and the number of outliers — the meaning of outlier is presented as a footnote in Table 2).

3.1. Comparison of different MLP architectures and training algorithms

A standard technique in neural networks is to train the network using one set of data, but to check performance against a validation set not used in training. Without validation, a network with a large number of weights can overfit the training data — loosing the underlying structure. The ability of a network not only to learn the training data, but to perform well on previously-unseen data, is known as generalization. During training although the training error decreases almost to zero, the validation error first decreases and then begins to rise again (Fig. 2). This is a sure sign that overlearning is occurring, and training should stop once deterioration in the validation error is observed (epoch 12 in the example of Fig. 2). Further to the validation data set, strictly speaking, it is highly recommended the examination of network's performance against a third set of data which has not been used in both the training and validation process. This is the test set. To this end, the 25 compounds were divided into three data sets namely the training set of 15 compounds, the validation set of 5 compounds and the test set of 5 compounds to examine the predictive ability of the examined MLP networks. The main requirement during training is the data representativity, meaning that the samples in the data set should be (evenly) spread over the expected range of data variability. In order to avoid the risk of not selected representative samples during training, as happens with random selections, we performed (results not shown) two different strategies of training set design suggested by Wu et al. [23] and Simon et al. [24], namely the D-optimal design (maximizing the determinant of the information matrix $|\boldsymbol{X}'\boldsymbol{X}|$) and the Kohonen self-organizing map approach (the main goal is to map objects from *n*-dimensional into two-dimensional space). Applying these methods, the samples chosen were spread in the whole space. Having defined the number of inputs and outputs, the number of hidden layers and the nodes in each layer the networks were ready for training. Each network was simulated 30 times with different initializations of the weights and the resulting networks were tested against the 5 unseen compounds (test set). Once the best network

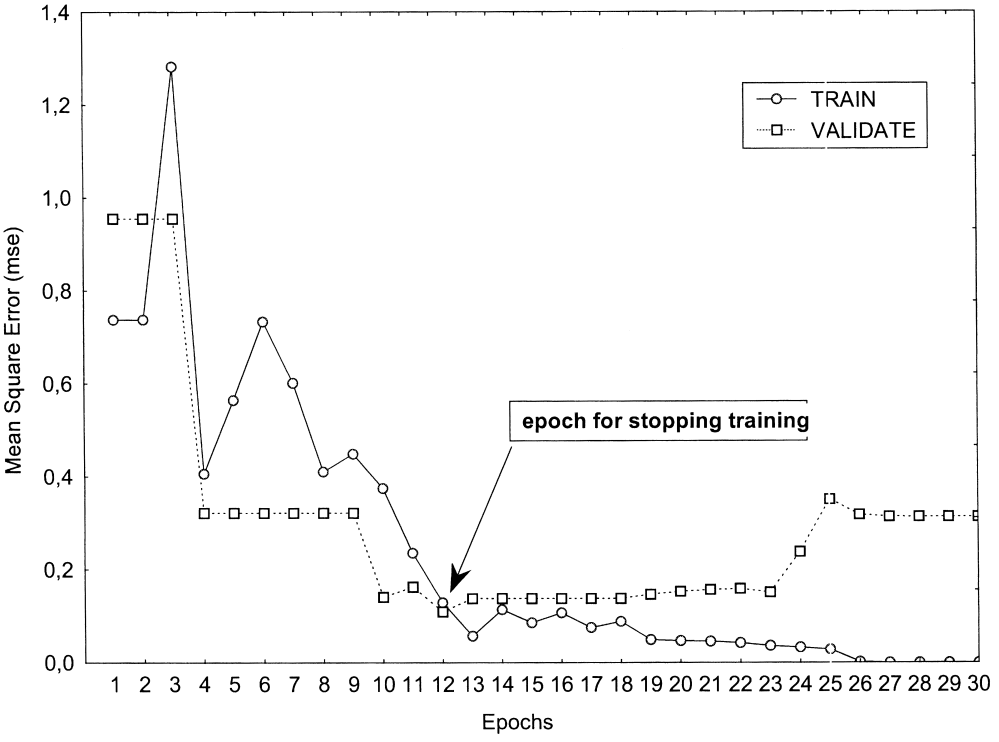

Fig. 2. Training and validation error diagrams during the training procedure.

was determined it was examined further by scrambling the input variables in order to examine any chance effect in the prediction ability of the isolated network. The input scrambling did not deteriorate network's performance indicating robustness in its prediction. Finally, in order to identify possible chance effects, another test was performed. During this test the matrix of independent variables was kept constant while the vector of dependent variables was randomly scrambled running the same algorithms described above. The same procedure was repeated several times and the results obtained were significantly inferior (higher RMSEP) to the previous ones, indicating that the results with the real data were not obtained due to chance effects.

It should also be mentioned that from a selection of different activation functions the best results were obtained using a sigmoid transfer function in the hidden layer and a linear function in the output layer. With sigmoidal hidden units, the universal approximation properties of the network hold even if the output units have linear activation functions. Generally, for interpolation problems, where we wish to generate mappings whose outputs represent smoothly varying quantities, it is convenient and sufficient to choose the output unit activation functions to be linear. The selection of the number of hidden neurons is usually done by trial and error. The goal is to find a network with a minimum number of neuron connections that are able to solve the problem. Networks with many connections can learn a problem perfectly, but their predictive ability may be very low. A network with one hidden layer of eight neurons was found to be sufficient to solve the present problem of predicting the log k' of aromatic solutes. The final step in the design of the MLP networks was the selection of the training algorithms. It is obvious from the current study that the algorithm selection is of major importance concerning the networks' predictive ability.

The examined algorithms are presented in Table 2 together with the training error (MSE), the post

training correlation coefficient R and the number of outliers. The algorithms examined in the present study range from the problematic steepest descent with momentum and learning rate up to the BFGS and LM second order algorithms, which converge rapidly, and they do not need the presence of any adjusting parameters such as momentum and learning rate. It is surprising the gain in network performance using some of these algorithms, and especially the LM, meaning time to converge and number of outliers (Table 2). The LM algorithm in most initializations converges almost after 20 epochs without any further error improvement in the following epochs (up to 1000). Also the conjugate gradient algorithm showed almost the same behavior but with lower degree of accuracy in its prediction. In Table 2 the algorithms are sorted according to their efficacy, from the higher to lower predictive ability. The performance of a trained network can be measured to some extent by the errors on the training, validation and test subsets, but in the present study we investigated the network response in more detail. To this end a regression analysis between the network responses in the whole dataset and their corresponding observed values was performed. This study returns the correlation coefficient (R value) between the predicted and observed values, a measure of how well the ANN fits the dataset and generalizes, meaning its predictive ability in new (unseen) datasets. Fig. 3 shows the linear relationships between the calculated values for the log k' of the first stationary phase, the predicted following the trained ANN with the LM algorithms and the PLS methods and the published ones. The same relationships (not shown) were resulted by the second stationary phase.

From Table 2 it is becoming evident that the LM algorithm resulted in the best performance with zero outliers in five new (unseen) solutes and the highest correlation coefficient of 0.996. Increasing the number of hidden neurons the correlation coefficient was decreased with an increase in the number of outliers (overtraining). Generally, the primary indicator for choosing the network architecture is the number of outliers in unseen (new) data. Polak-Ribiere conjugate gradient algorithm (PRCG) follows the performance of LM algorithm resulting in one outlier in the test data set. One outlier resulted also from the BFGS and scaled conjugate gradient algorithms

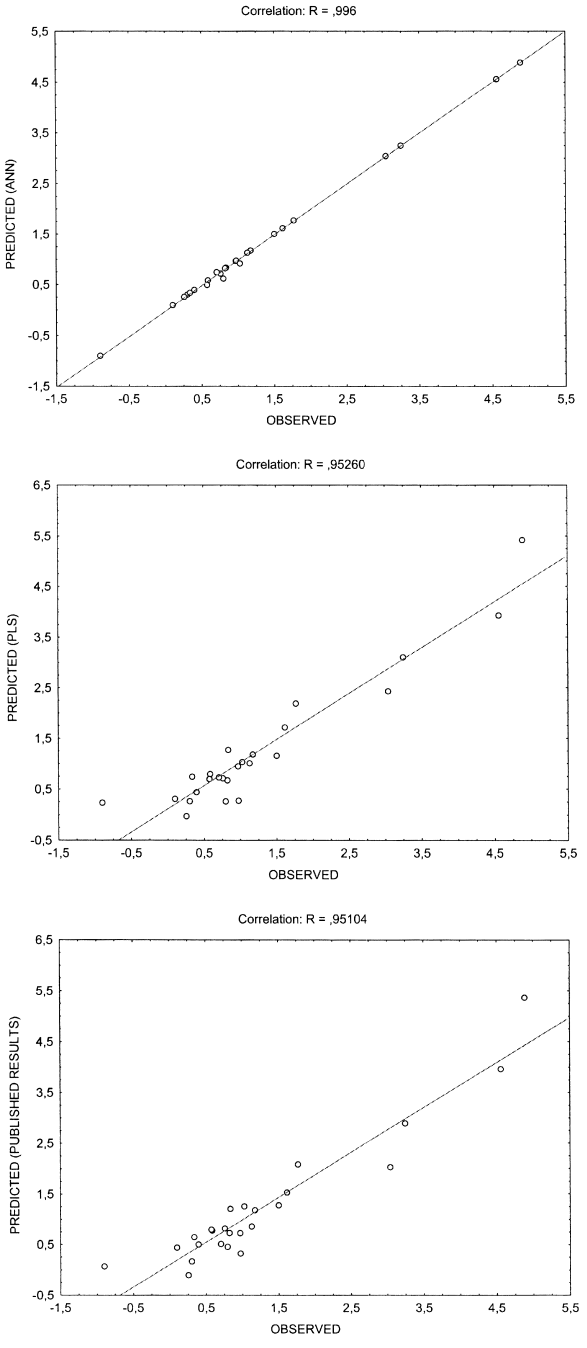

Fig. 3. Linear fit for the calculated (experimental) log k' values against the predicted ones from the ANN and PLS models and the published ones.

while the predicted values in the unseen data set were in both cases less accurate than the ones derived from the PRCG algorithm. With three outliers follow two conjugate gradient algorithms (Powl-Beale and Fletcher-Powell). Finally, the worst performance with four and five outliers was seen from the different types of backpropagation algorithms (see Table 2). Further to the higher prediction ability of the algorithms (LM and PRCG) the time for training was decreased substantially to less than a minute.

Having established a configuration for the neural network, a testing procedure was carried out. During this process five compounds were removed from the training data sets and served as test set. After training, the parameters of the compound unknown to the network were put into the network and the predictive activities of these compounds were obtained (Table 3). From the values in Table 3 it is becoming evident the superior predictive ability of ANN (with the lowest prediction error) compared to that of the published results and to that from the PLS procedure (see later). The ANN predictive values differed by at most 0.18 leading to even higher accuracy and providing reliable predictions.

It is becoming evident that it is difficult to know 'a priori' which training algorithm will be the best and the fastest for a given problem. It depends on many factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network and the error goal. In general, the LM algorithm converges fast and this advantage is especially noticeable if very accurate training is required, as in the present study. Of the conjugate gradient algorithms, the Powell-Beale procedure requires the most storage, but usually has the fastest convergence. The quasi-Newton methods are often the next fastest algorithms; the BFGS algorithm does require storage of the approximate Hessian matrix, but is generally faster than the conjugate gradient algorithms. The variable learning rate algorithm is usually much slower than the other methods but it can still be useful for comparison reasons or in cases which require slow convergence.

3.2. Partial least squares regression

PLS was applied to our example problem of developing a model that relates the log k' to the eight descriptors. As before, we will divide the 25 samples available in training validation and test subsets as before. Next to the division of dataset, PLS model attempts to maximize covariance (to do both, capture variance and achieve correlation). The question that arises is: how many factors (latent variables) should be chosen for maximum covariance? Either the rule of thumb that one should choose additional factors when the predicted residual sum of squares (PRESS) for the test subset improves by at least 2% could be used, or thr graphic representation of the behaviour of the root-mean-square error of cross-validation (RMSECV) which is a measure of the model's ability to predict new samples. Fig. 4 shows the graphical representation of the number of factors against RMSECV and RMSEC (root-mean-square error of calibration; it tells us about the fitting of the model to the calibration data). From Fig. 4 it is becoming evident that the model with 4 latent variables performs the best prediction. In latent variables 5–8 even if the RMSEC continues to decrease there is a significant increase in RMSECV. The development of a PLS model with 4 factors

Table 3
Test set of five solutes, the observed (experimental) log k', the predicted from the network trained with the LM algorithm the predicted from the PLS method and the published ones [4]

Solutes	Experimental	ANN predicted	PLS predicted	Published
1	0.758 (0.758)[a]	0.718 (0.706)	0.712 (0.655)	0.818 (0.633)
2	1.027 (0.829)	0.918 (0.905)	1.230 (1.232)	1.256 (1.112)
3	0.797 (0.252)	0.618 (0.365)	0.261 (0.298)	0.455 (0.296)
4	0.574 (−0.021)	0.495 (−0.156)	0.850 (0.196)	0.798 (0.648)
5	0.705 (0.337)	0.745 (0.398)	0.729 (0.389)	0.512 (0.267)
RMSEP	–	0.103 (0.092)	0.254 (0.211)	0.229 (0.331)

[a] Numbers in parenthesis correspond to log k' with the second stationary phase.

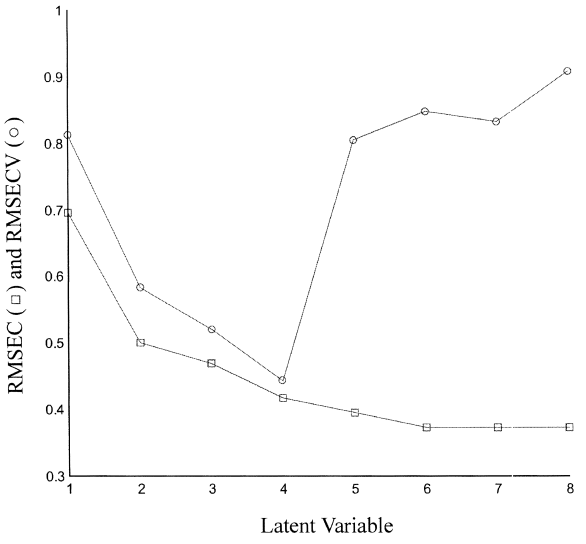

Fig. 4. RMSEC and RMSECV vs. latent variables in PLS training and validation procedure.

results the predictions shown in Table 3. In Table 3 the root-mean-square error of predictions (RMSEP) is also presented when the examined models are applied to new data provided that the reference (calculated) values for the new data are known. RMSEP is calculated as follows:

$$\text{RMSEP} = \sqrt{\frac{\sum_{1}^{n} |y_{\text{obs}} - y_{\text{pred}}|^2}{n}}$$

The superior performance of ANN model compared to that of PLS and the published results is evident from the RMSEP values.

4. Conclusion

The data set of the 25 substances as solutes to different stationary phases is a representative sample from the population of solute–stationary phase interactions where it is necessary to model the interaction procedure and to predict the log k' values. The algorithms for training the feed-forward networks such as the LM or PRCG improved the predictions for unseen data and the networks reached a significant level of accuracy outperformed both the published results and the results from the PLS regression. Is the present set of the eight descriptors (inputs) the most representative (predictive) for the examined case? This question could not be answered straightforwardly since the extraction of the significant variables, further to the examination of possible strong relationships, requires the development of methods such as genetic algorithms, which will be described in detail in a forthcoming study. In any ANN study the accuracy could probably be increased further by performing input preprocessing, meaning to find a more representative group of inputs.

References

[1] M. Balcan, T. Cserhati, E. Forgacs, D.F. Anghel, Biomed. Chromatogr. 13 (1999) 225.
[2] E. Forgacs, T. Cserhati, J. Pharm. Biomed. Anal. 18 (1998) 505.
[3] C. Bishop, Neural Networks for Pattern Recognition, University Press, Oxford, 1995.
[4] T. Cserhati, E. Forgacs, K. Payer, P. Haber, R. Kaliszan, A. Nasal, LC–GC Int. 11 (1998) 240.
[5] D.L. Massart, B.G.M. Vandeginste, L.M.C. Buydens, S. de Jong, P.J. Lewi, J. Smeyers-Verbeke (Eds.), Handbook of Chemometrics and Qualimetrics: Part B, Elsevier, Amsterdam, 1998, Chapter 35.
[6] L. Fausett, Fundamentals of Neural Networks, Prentice Hall, New York, 1994.
[7] J. Zupan, J. Gasteiger, Neural Networks For Chemists, VCH Verlagsgesellschaft, Weinheim, 1993.
[8] G.C. Looney, Pattern recognition using neural networks, Oxford University Press, New York, 1997.
[9] D.E. Rumelhart, J.L. McClelland, Parallel Distributed Processing. Experiments in the Microstructure of Cognition, MIT Press, Cambridge, MA, 1986.
[10] M.C. Nelson, W.T. Illingworth, A Practical Guide To Neural Nets, Addison-Wesley, Reading, MA, 1991.
[11] S. Lawrence, G.L. Giles, A.C. Tsoi, What Size Neural Networks Gives Optimal Generalization? Convergence Properties of Backpropagation. Technical report, UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University Maryland, College Park, MD, 1996.
[12] W.S. Sarle, Stopped training and other remedies for overfitting, in: Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, 1995, p. 352.
[13] A. Weigend, On overfitting and the effective number of hidden units, in: Proceedings of the 1993 Connectionist Models Summer School, 1994, p. 335.
[14] D. Nguyen, B. Widrow, Proceedings of the International Joint Conference on Neural Networks, 3 (1990) 21.
[15] C. Charalambous, IEEE Proc. 139 (1992) 301.

[16] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[17] J.C. Principe, N.R. Euliano, W.C. Lefebvre, Neural and Adaptive Systems: Fundamentals Through Simulations, Wiley, 2000.
[18] K. Levenberg, Quart. J. Appl. Math. II 2 (1944) 164.
[19] D.W. Marquardt, J. Soc. Indust. Appl. Math. 11 (1963) 431.
[20] Y.L. Loukas, Anal. Chem. 70 (1998) 966.
[21] Y.L. Loukas, J. Phys. Chem. B 101 (1997) 4863.
[22] Matlab version 5.2, MathWorks 24 Prime Park Way, Natick, MA.
[23] W. Wu, B. Walczak, D.L. Massart, S. Heuerding, E. Erni, I.R. Last, K.A. Prebble, Chem. Intell. Lab. Syst. 33 (1996) 35.
[24] V. Simon, J. Gasteiger, J. Zupan, J. Am. Chem. Soc. 115 (1993) 9148.