Neural Networks Convergence Using Physicochemical Data

Mati Karelson,*[,†,‡] Dimitar A. Dobchev,[†,§] Oleksandr V. Kulshyn,[†,§] and Alan R. Katritzky[§]

Department of Chemistry, University of Tartu, 2 Jakobi Street, Tartu 51014, Estonia, Department of Chemistry, Tallinn University of Technology, Ehitajate tee 5, Tallinn 19086, Estonia, and Center for Heterocyclic Compounds, Department of Chemistry, University of Florida, Gainesville, Florida 32611

Received January 18, 2006

An investigation of the neural network convergence and prediction based on three optimization algorithms, namely, Levenberg−Marquardt, conjugate gradient, and delta rule, is described. Several simulated neural networks built using the above three algorithms indicated that the Levenberg−Marquardt optimizer implemented as a back-propagation neural network converged faster than the other two algorithms and provides in most of the cases better prediction. These conclusions are based on eight physicochemical data sets, each with a significant number of compounds comparable to that usually used in the QSAR/QSPR modeling. The superiority of the Levenberg−Marquardt algorithm is revealed in terms of functional dependence of the change of the neural network weights with respect to the gradient of the error propagation as well as distribution of the weight values. The prediction of the models is assessed by the error of the validation sets not used in the training process.

1. INTRODUCTION

Neural networks have been applied in many diverse scientific endeavors ranging from economics, engineering, physics, and chemistry, to medical science, etc.[1] These computational methods evolved from attempts to understand and emulate the brain's information processing capability. The brain consists of multimodal neural networks that extract and recombine relevant information received from their environments and render the brain capable of making decisions that satisfy the needs of the organism. These capabilities of the brain can be emulated with artificial neural networks (ANN), which can conceive complex nonlinear input−output transformations. Their nonlinear feature extraction capability suggests their potential usefulness in Quantitative Structure−Property/Activity Relationship (QSPR/QSAR). However, applications to this area pose a special problem for ANN's: unlike the subjects of other applications, molecules lack an obvious numerical model.

In recent years, the literature concerning ANN as applied to QSPR/QSAR has grown drastically, suggesting that the importance of applications of ANN in molecular modeling is a major driving force. Among numerous studies using ANN in QSPR/QSAR, major contributions are due to the groups of Jurs,[2] Zupan and Gasteiger,[3] and Zefirov[4] and ourselves[5−7] among others. These authors have shown that the ANN models frequently possess better predictive characteristics compared to models using standard multilinear regressions. The flexibility of the ANN has better ability to represent predictive models.

Since the back-propagation learning algorithm[8] was first popularized, there has been considerable research on methods to accelerate the convergence of the algorithms. The first category involves the development of ad hoc techniques.[9] These techniques include such ideas as varying the learning rate, using momentum and rescaling variables. Another category of research has focused on standard numerical optimization techniques.[10]

The most popular approaches from the second category have used conjugate gradient or quasi-Newton methods. The quasi-Newton methods are considered to be more efficient, but their storage and computational requirements go up as the square of the size of the network. However, limited memory quasi-Newton algorithms speed up convergence while limiting memory requirements.[11] If exact line searches are used, the one step secant methods produce conjugate directions.

A technique of numerical optimization that has been applied to neural networks is nonlinear least squares.[12] The more general optimization algorithms were designed to work effectively on all sufficiently smooth objective functions. However, when the form of the objective function is known it is often possible to design more effective algorithms. One particular form of the objective function of interest to a neural network is a sum of the squares of nonlinear functions. A good algorithm for such objective functions is the Levenberg−Marquardt algorithm, which works better than the quasi-Newton and the conjugate algorithm for analytical functions.[13,14]

Investigating the convergence of the ANN for a certain problem can reduce computational time as well as increase the efficiency of the neural network. In addition, as the optimization algorithm improves, so does the minimization of the function. Many problems in chemistry lack adequate analytical tools for explanation or prediction. Neural networks may prove useful in certain problems that existing methods have solved, provided that the network solves the problem more rapidly or less expensively than conventional

* Corresponding author phone: +372 6 202 814; fax: +372 6 202 819; e-mail: mati.karelson@ttu.ee.
† University of Tartu.
‡ Tallinn University of Technology.
§ University of Florida.

10.1021/ci0600206 CCC: $33.50
Published on Web 09/02/2006

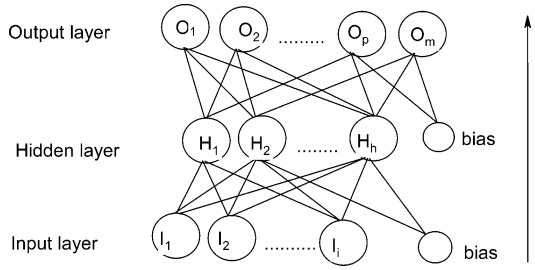

Figure 1. Three-layer back-propagation neural network.

methods. Although the process of training a network to predict a given property usually takes a long time, a trained network makes predictions very quickly—in some cases, fast enough to make new applications possible. The main goal of the present work is to investigate the convergence of the back-propagation neural network and to show to some extent that the prediction ability of the net using the Levenberg−Marquardt algorithm provides better results as compared with the conjugate gradient algorithm and delta rule applied on various physicochemical data.

2. METHODS

2.1. Back-Propagation Feed-Forward Neural Network. Artificial neural networks (ANNs) are composed of a number of single processing elements (PE) or units (nodes). Each PE has weighted inputs, transfer function, and one output. PEs are connected with coefficients (weights) and are organized in a layered topology as follows: (i) the input layer, (ii) the output layer, and (iii) the hidden layers between them. The number of layers and the number of units in each layer determine the functional complexity of the ANN.

In this work, a back-propagation fully connected network was developed and used to obtain a nonlinear QSAR/QSPR models. Topologically, it consists of input, hidden, and output layers of neurons or units connected by weights as shown in Figure 1. Each input layer node corresponds to a single independent variable (molecular descriptor) with the exception of the bias node. Similarly, each output layer node corresponds to a different dependent variable (property under investigation). For more detailed discussion see, for example, ref 15.

Associated with each node is an internal state designated by I_i, H_h, and O_m for the input, hidden, and output layers, respectively. Each of the input and hidden layers has an additional unit, termed a bias unit, whose internal state is assigned a value of 1.

Training of the neural network is achieved by minimizing an error function E with respect to the bond weights $\{w_{hi}, W_{mh}\}$

$$E = \sum_p E_p = \frac{1}{2}\sum_p \sum_m (a_{pm} - O_{pm})^2 \qquad (1)$$

where E_p is the error of the pth training pattern, defined as the set of descriptors and activity corresponding to the pth data points, or chemical compound; a_{pm} corresponds to the experimentally measured value of the mth dependent variable, in our case there were eight different properties.

One of the standard algorithms for minimizing E is the delta rule.[15] The algorithm is based on an iterative procedure for updating the weights of the neural network from their initially assigned random values.

2.2. Levenberg−Marquardt Optimizer. This section describes briefly the Levenberg−Marquardt algorithm and the conjugate gradient algorithm. These algorithms are well-known, and for a more detailed description the reader can refer to ref 16.

Since back-propagation utilizes the descent algorithm, the Levenberg-Marqurdt algorithm[17] is an approximation of so-called Newton's methods. The Levenberg-Marqurdt algorithm is a gradient-based technique, capable of determining search directions according to an objective function's derivative information. In fact, the Levenberg−Marquardt approach is one of the main algorithms used for neural network learning in conjunction with back-error propagating process.

We focus on minimizing a real-valued objective function E: $R^n \rightarrow R$ defined as an n-dimensional input space $\theta = [\theta_1, \theta_2 ..... \theta_n]^T$ consisting of the connection weights. Finding a (possibly local) minimum point $\theta = \theta^*$ that minimizes $E(\theta)$ is of primary concern. Due to the possible complexity of E we often resort to an iterative algorithm to explore the input space efficiently. In iterative descent methods, the next point θ_{k+1} is determined by a step down from the current point θ_k in a direction vector **d**

$$\theta_{k+1} = \theta_k + \alpha_k \mathbf{d}_k \quad (k = 1, 2, ...) \qquad (2)$$

where k denotes the current iteration number, and θ_{k+1} and θ_k represent two consecutive elements in a generated sequence in solution candidates $\{\theta_k\}$.

The key role in the derivative-based methods (such as Levenberg−Marquardt) is the determination of descent direction d. It can be defined by using the first and second derivatives of the objective function E with respect to the weights or so-called Jacobian and Hessian matrices $\boldsymbol{J}$ and $\boldsymbol{H}$. In the latter the second derivatives can be neglected as small contributors of its expansion.[18] Thus, the Levenberg−Marquardt optimizer has the general form (3)

$$\theta_{k+1} - \theta_k = \boldsymbol{J}^T(\theta)\boldsymbol{J}(\theta) + \xi \boldsymbol{I}^{-1}\, \mathbf{J}^T(\theta)\mathbf{e}(\theta) \qquad (3)$$

where $\boldsymbol{I}$ is the unity matrix and $\boldsymbol{e}$ is the error for the kth input of the network. The scalar parameter ξ is multiplied by some factor (ν) defining the step in the iterative procedure of the optimization. In other words, whenever a step reduces (1), ξ is divided by ν. In the next section we used $\xi = 0.01$ as a starting point with $\nu = 10$. At this point, it should be mentioned that when ξ is large the algorithm becomes steepest descent (using step $1/\xi$), while for small ξ the algorithm becomes Gauss−Newton (i.e. $\xi \boldsymbol{I} = 0$ in (3)).

Conjugate direction methods are also based on the second-order approximation of the objective function E as the Levenberg−Marquardt algorithm. The general difference between these to algorithms is how $\mathbf{d}_k$ in (2) is updated. We used in this work the Polak−Ribiere modification[16] of the conjugate gradient method.

2.3. Summary. The main computational effort in Levenberg−Marquardt and conjugate gradient algorithms is the calculation of the Jacobians. For the neural network problem the terms in the Jacobian can be computed by using the standard back-propagation algorithm with one modification at the final layer. In the error back-propagation algorithm,

the weight updates are proportional to the error propagating from the output through the derivatives of activation function and through the weights. This is a consequence of using the steepest descent method for calculating the weight adjustments. Convergence properties of the learning process will depend on the algorithm used that finds better adjustments to the weights.

In this work we used an in-house program based on the formulas provided in this section. The program was written in C++ language with double float precision. C++ language has advantages to make abstraction of the data provided and separated in several classes (e.g. class for neurons, class, architecture of the ANN, etc.). For solving some systems of equations concerning the Jacobian matrix we used modification of the QR-decomposition algorithm. In addition, several pseudorandomization routines were implemented for the generation of random numbers.[19]

3. RESULTS

The Levenberg−Marquardt algorithm applied in the back-propagation neural network as described in section 2.2 was tested for fast convergence on eight physicochemical data sets. To provide a basis of comparison, two other modifications to the back-propagation were also implemented for the same problem, namely the conjugate gradient algorithm and the simple delta rule optimization. For the purposes of this investigation we used the Polak−Ribiere version of the conjugate gradient algorithm.[16] This version of the method is known to be able to handle most of the practical problems as well as for its ability to find better minimum for the object function.

Eight neural network models were built for each of the physicochemical data sets using the respective Levenberg−Marquardt (L), conjugate gradient (C), and delta rule (D) algorithms. These data sets were taken from the following sources.[21−28] For five of them, the ANN methodology has been previously applied.

In QSAR/QSPR modeling the neural network models possess certain architecture with input neurons in the range 2−12 and hidden layers from 1 to 3.[20] Consequently, the main goal of this work is to simulate a practical neural network structure that usually is used in QSAR/QSPR modeling and to explore this structure for fast convergence and prediction ability. For all neural network simulations (batch training with random shuffling) the molecular descriptors used as inputs were calculated for all chemical compounds by Codessa Pro software.[21] The descriptors were selected by the Heuristic method implemented in Codessa Pro[22] as those appearing in the respective best multilinear equations.

3.1. ANN Simulation of Physicochemical Data for 411 Vapor Pressures. The first problem was to test the convergence of the three algorithms described in the previous section for 411 organic compounds with experimental values for the vapor pressure (logVP). The data set was taken from ref 23. This data set includes all the compounds used for building the multilinear QSPR model. All molecular structures were optimized using the AM1 method. The descriptors were calculated based on these optimized structures by Codessa Pro. For the selection of the descriptors as input units to the neural network, the best six-parameter multilinear equation obtained by the Heuristic method[21] was used. These descriptors are *Gravitation* index (all bonds), Relative number of benzene rings, FNSA-2 Fractional PNSA (PNSA-2/TMSA) (MOPAC PC), Square root of Charged Surface Area for atom C, HA dependent HDCA-1 (Zefirov PC), and Number of F atoms. The simulated neural network had then 6−6−1 architecture that is a usual architecture in the nonlinear QSPR modeling. We also investigated several other ANN models with different architecture (5−6−1, 7−6−1, and 7−5−1), and according to the lowest root-mean-squared error (RMS) (averaged from the results of the three algorithms for the respective four ANN architectures) the topology 6−6−1 was selected.

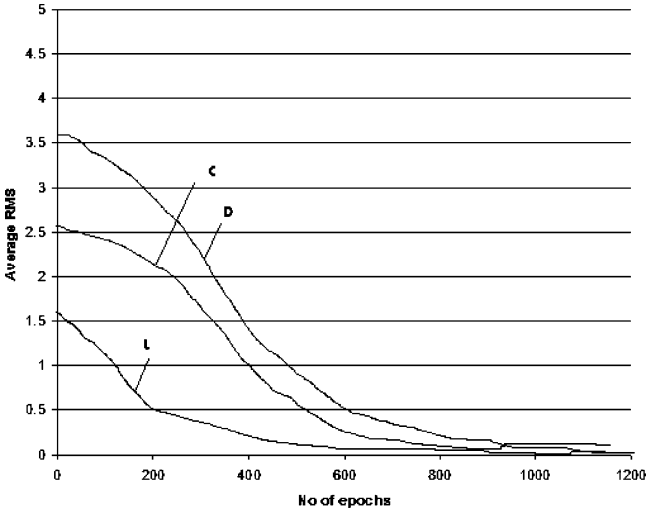

Figure 2. Network (6−6−1) convergence for 411 compounds with measured VP.

Before the start of training of the ANN, the whole data set (411 compounds) was divided into two parts: (i) training set (4/5 from the whole set, 329 compounds) and (ii) validation set (1/5 from the whole set, 82 compounds). The validation set was not used in the training process except for monitoring of the validation RMS to avoid overfitting of the model. Thus, the RMS was used as a stopping criterion for the training when it started to increase.

In Figure 2 are given the results from this neural network simulation. This plot displays the training curves of the averaged (by loops, to smooth the curves) RMS for the three methods.

3.2. ANN Modeling for 298 Boiling Points. The data for the second simulation were taken from ref 21. In this case, the optimum neural network that was built for the ANN exploration consisted of the following architecture 5−5−1 with input descriptors *Gravitational index* (all bonds), Highest normal mode vibration transition dipole, FPSA-3, Minimum atomic charge for atom C, and Highest normal mode vibration frequency. Again, we chose the optimum ANN topology from several architectures as was described in the previous subsection. Figure 3 gives the results for the convergences from the training of the network. As seen from this figure, the Levenberg−Marquardt algorithm again converged faster than other optimization algorithms.

3.3. Physicochemical Data for 115 Milk/Plasma Ratios. Figure 4 illustrates the training curves of the third test problem in which a 4−4−1 neural network was built based on 115 milk/plasma concentration ratios. The experimental

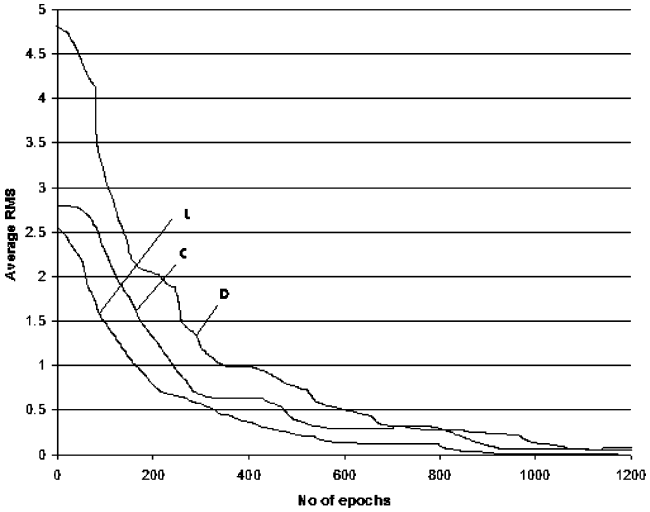

Figure 3. Network (5−5−1) convergence for 298 compounds with measured boiling points.

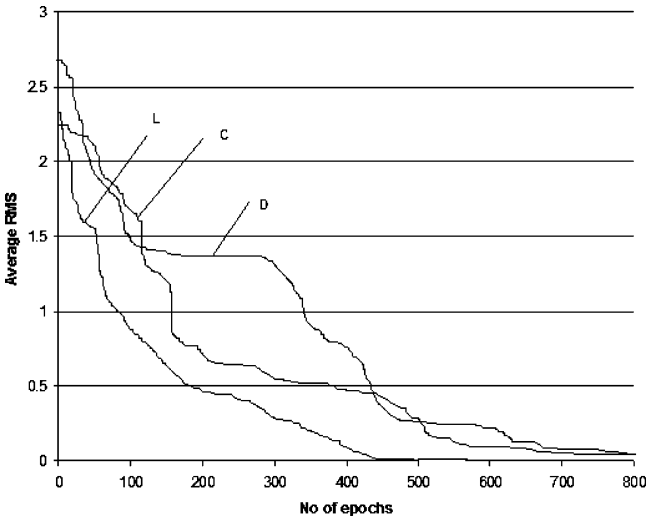

Figure 4. Network (4−4−1) convergence for 115 compounds with measured MP ratios.

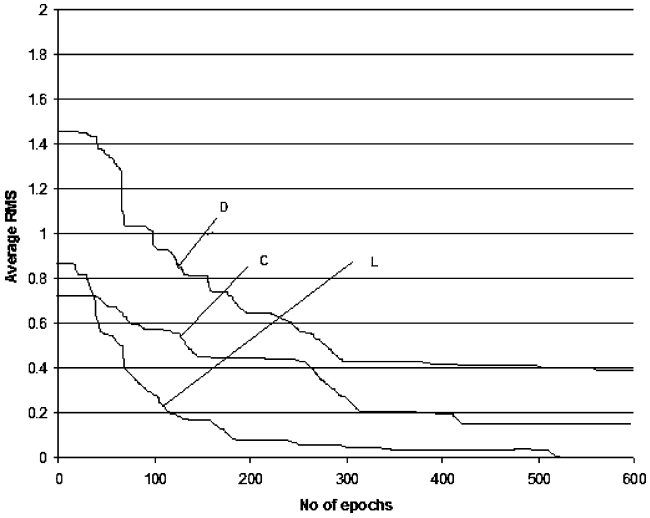

Figure 5. Network (5−4−1) convergence for 60 compounds with measured carcinogenic affinities.

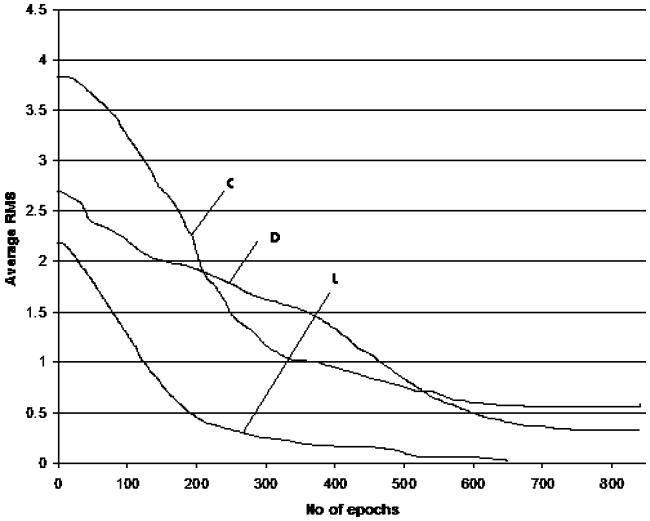

Figure 6. ANN (5−4−1) convergence results for 115 compounds with measured log BBB.

data values and the compounds were taken from ref 24 .The four descriptors obtained by the Heuristic method of Codessa Pro were as follows: *Total enthalpy (300K)/natoms*, RNCS charged SA, Number of F atoms, and *1X Beta polarizability*.

3.4. Physicochemical Data for 60 Carcinogenic Activities. The same data for 60 compounds with experimental carcinogenic activities were used in this work as in a previous study of an ANN modeling.[25] According to the Heuristic method it was found that five descriptors were statistically significant for building an ANN model with architecture 5−4−1. These descriptors were *Gravitation Index (all bonds)*, *HA Dependent HDSA-2/SQRT(TMSA) (Mopac)*, *DPSA3 Difference in CPSAs* (PPSA3-PNSA3) (Zefirov PC), Kier shape index (order 2), and Maximum Coulombic interaction for bond C−C. Again, the data set was divided into two sets, training and validation, and we applied the same methodology as described in subsection 3.1. In Figure 5 are given the convergence results for this ANN model.

3.5. Physicochemical Data for 115 logBBB. The data for 115 organic compounds with measured blood-brain partition coefficient log(BBB) were taken from ref 25 and were also used in ANN treatment. For these data, the Heuristic method implemented in Codessa Pro obtained a 5-descriptor multilinear equation that was the best among six other equations. Thus, the descriptors of this equation were served as inputs to the ANN model. These were *Charged Surface Area (MOPAC PC) for atom* C, DPSA2 Difference in CPSAs (PPSA2-PNSA2) (Zefirov PC), Kier flexibility index, Max 1-electron react. index for atom C, and HACA H-acceptors charged surface area (MOPAC PC).

During the building of the neural network architecture, we tried to obtain good generality[15] of prediction for the network (as in the case of all ANN models developed herein). More precisely, we tried to derive as small a number of processing units (i.e. not to overparametrize the model) in the network as possible but simultaneously to have a good predictivity for the validation set. Thus, among several neural network topologies we found that the best model showing the lowest average RMS consists of 5−4−1 architecture. The convergence results of this model are shown in Figure 6. As can be seen from the figure, the Levenberg−Marquardt optimizer implemented in this model converged faster than others as in the case of all ANN models built so far.

3.6. Data for 137 Organic Compounds with Measured Ozone Tropospheric Degradation Rate. The data for 137

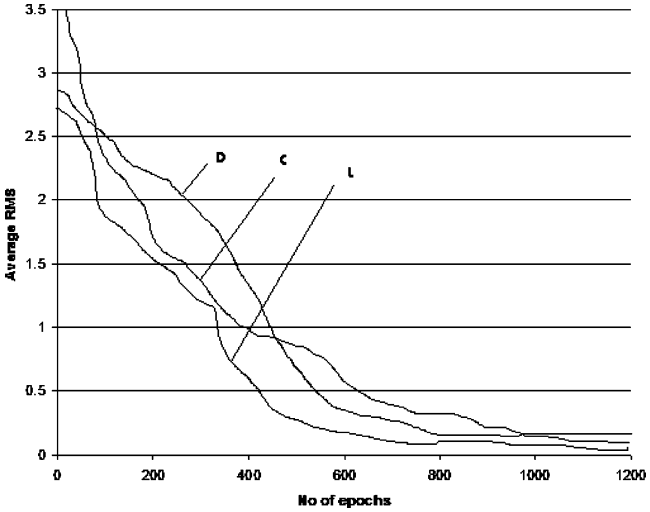

Figure 7. ANN (6−6−1) convergence results for 137 compounds with measured ozone tropospheric degradation rates (log KO_3).

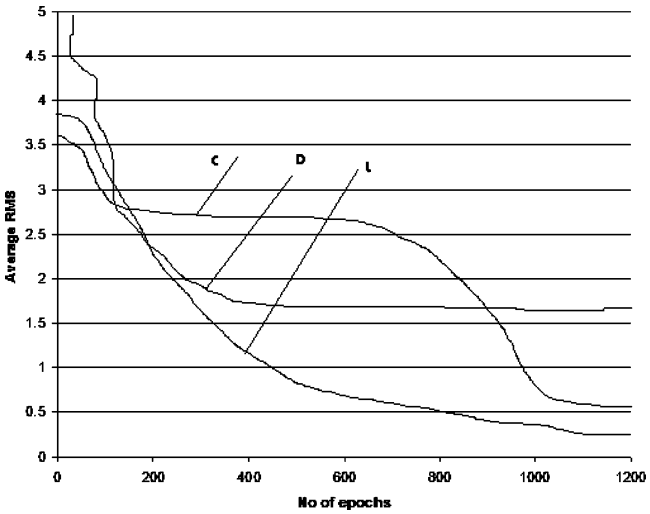

Figure 8. ANN (7−5−1) convergence results for 158 compounds with measured skin permeation rates (log Kp).

ozone tropospheric degradation rates (log KO_3) were taken from ref 26. In this work, the experimental values of log KO_3 were used in multilinear and ANN treatment to build predictive QSPR models showing a superiority of the ANN prediction over the multilinear models. Our ANN model resulted in 6-descriptor inputs with the following architecture 6−6−1. These descriptors are the *Minimum bond order for atom C,* HOMO-1, Kier and Hall index, H donors FPSA2, Maximum electron reaction index for atom C, and Maximum repulsion for bond C−H. The convergence results for the training set are given in Figure 7. In this ANN model one of the descriptors is the same as in the work[26] i.e., *Minimum bond order for atom C*. Also, *HOMO-1* is close to the therein[26] found descriptor *HOMO−LUMO* showing similarity of the ANN models.

3.7. Physicochemical Data for 158 Skin Permeation Rates. In this simulation and modeling we used 158 experimental skin permeation rates of organic compounds taken from ref 27. The developed optimum ANN model had architecture 7−5−1. The convergence results for the training set are shown in Figure 8. The seven descriptors for this model chosen by the Heuristic method in Codessa Pro were *Kier flexibility index, Max 1-electron react. index for atom C, HACA H-acceptors charged surface area (Zefirov* PC), HBSA H-bonding surface area (MOPAC PC), Bonding Information content (order 1), Charged Surface Area (MOPAC PC) for atom C, and Partial Surface Area for atom H.

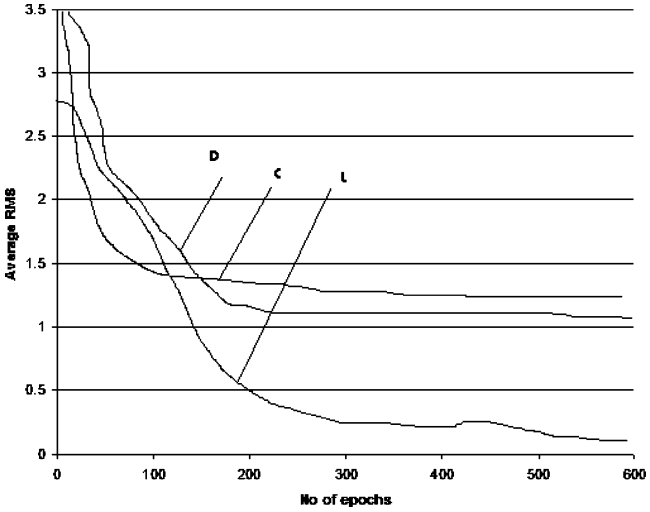

Figure 9. ANN (4−4−1) convergence results for P-glycoprotein inhibitors for 57 compounds (log Kd).

3.8. Flavonoids as P-Glycoprotein Inhibitors for 57 Compounds. The last data set that was subjected to the ANN modeling consisted of 57 experimental binding affinities log Kd of organic compounds taken from ref 28. The modeling procedure resulted in 4−4−1 ANN. The four inputs of the network were *HA dependent HDCA-2/SQRT(TMSA), Minimum coulumbic interaction for bond C−H, Square* root of surface area of atom C, and Total molecular electrostatic interactions and XY shadow/XY rectangle. [All descriptor definitions in this work can be found in ref 29.] The convergence results of this model are shown in Figure 9.

4. DISCUSSION

The results from the above section clearly show that the fastest algorithm implemented in the back-propagation neural network is the Levenberg−Marquardt. Also, from Figures 4−9 it can be seen that the starting average RMS in most of the cases for the Levenberg−Marquardt algorithm is less than the other two algorithms. We used in these figure the averaged RMS in order to smooth the curves and to escape the sharp picks of the actual RMS for more obvious trends.

In two of the cases (the models for ozone tropospheric degradation rates and P-glycoprotein inhibitors, see Figures 8 and 9), the Conjugate gradient and the delta rule algorithms failed to converge, whereas the Levenberg−Marquardt algorithm converged sufficiently fast. We performed three separate ANN trainings for these two data sets, and the results remained invariants. The statistical results for all models are shown in Table 1. As can be noted from the table for the first data set (vapor pressure) the stopping validation RMS's for the three algorithms L, C, and D were 0.412, 0.483, and 0.562, respectively. Consequently, the least RMS of the Levenberg−Marquardt algorithm indicates better predictive ability of the vapor pressure for this ANN model. In addition, it can be seen from Figure 2 and Table 1 that the Levenberg−Marquardt (L) algorithm converged after 591 epochs (RMS = 0.101) being therefore faster than the conjugate gradient (C) and the delta rule (D) algorithms 635 (0.340) and 711

Table 1. Statistical Results for All ANN Models

no. of ANN model[a]	training RMS			validation RMS			epochs			absolute mean error (validation set)			S^2 (validation set)		
	L	C	D	L	C	D	L	C	D	L	C	D	L	C	D
1	0.101	0.340	0.398	0.412	0.483	0.562	591	635	711	0.081	0.130	0.197	0.138	0.177	0.189
2	0.198	0.293	0.312	0.342	0.395	0.431	795	948	828	8	10	13	156	184	173
3	0.296	0.411	0.563	0.522	0.595	0.621	401	487	948	0.142	0.172	0.201	0.134	0.156	0.197
4	0.096	0.211	0.413	0.222	0.295	0.321	293	338	426	0.088	0.119	0.191	0.143	0.231	0.254
5	0.218	0.284	0.301	0.276	0.312	0.384	325	453	511	0.114	0.210	0.244	0.350	0.388	0.391
6	0.399	0.451	0.488	0.482	0.551	0.612	654	721	734	0.240	0.318	0.452	1.12	1.54	1.661
7	0.482	0.668	0.589	0.592	0.770	0.655	865	1089	>1200	1.421	1.580	1.890	1.921	2.03	1.990
8	0.125			0.159			541			0.099			0.155		

[a] 1- model for vapor pressure, 2 − model for boiling points, 3 − model for milk/plasma ratios, 4 − model for carcinogenity data, 5 − model for blood-brain barrier partition coefficient, 6 − model for tropospheric degradation rates, 7 − model for skin permeation rates, 8 − binding affinities.

(0.398), respectively. All the results in the table indicate that in most of the cases the Levenberg−Marquardt algorithm provides better prediction ability, by comparing the absolute mean errors of the predicted validation sets as well as their standard deviations (last column of Table 1). Also, for model no. 8, both Conjugate gradient and delta rule algorithms failed to converge to small RMS. According to the RMS error of the validation set and predictive ability, those ANN models were weak. In addition, for model 7 the delta rule could not converge to a reasonably small error. It took more than 1200 epochs to fall into a local minimum of the error function producing a flat plateau (see Figure 8). Also, it can be noted that the conjugate gradient training led to a flat trend between 180 and 690 epochs. However, the Levenberg−Marquardt algorithm shows smooth convergence by reaching faster the stopping RMS of the validation error at 0.592 that is the smallest for these algorithms used.

All validation sets were chosen as containing 1/5 of the main data sets. They were not used in the training process of the ANN models. Generally, the prediction abilities of these models using the Levenberg−Marquardt algorithm produced smaller absolute mean errors as well as standard deviations. We had noted that for an ANN model with a larger number of processing units (larger number of connection weights) applied to a larger number of data sets leads in some cases to a slower convergence of the Levenberg−Marquardt algorithm as compared to the delta rule algorithm but faster convergences than the conjugate gradient. A possible reason for this is that the delta rule does not require calculation of the so-called Jacobian matrix of the derivatives of the errors. Five experimental calculations were done for the vapor pressure data in order to prove that two out of five runs in the Levenberg−Marquardt algorithm converged faster than the delta rule method. However, the prediction of the validation set (according to the mean error and the standard deviation of the validation set) is better than the delta rule method.

The graph in Figure 6 demonstrates that the delta rule implementation of the ANN converged faster than the conjugate gradient according to the RMS of the training set after 534 epochs. Generally, the results indicated that the conjugate gradient method is faster than the simple delta rule. In addition, during model building and ANN exploration for the log BBB data it was noted that the conjugate gradient method in some of the cases (one out of five) with a larger number of processing (7−7−1) units obtains better predictive results for the validation set than the L algorithm. However, the results in Table 1 show that for a real QSAR the generalization of the prediction power of the ANN is important, and in this case we have better predictive ability for the L algorithm.

As a rule the larger number of connection weights in an ANN, the slower is the training process. We noted that the prediction of the ANN occurs when the distribution of the net's weights keep a normal distribution shape. When the RMS of the validation set as the number of epochs increases, the distribution of the weights deviates strongly from normal distribution.

5. CONCLUSIONS

We have critically examined the ANN convergence using three different optimization algorithms i.e., Levenberg−Marquardt, conjugate gradient, and delta rule that were implemented in a fully connected back-propagation neural network. These algorithms are often used in the QSAR/QSPR modeling and prediction. In the present work, we simulated the building of eight ANN models based on a large number of compounds and a variety of their physicochemical and biomedical properties and analyzed the predictive power of these models regarding the different optimizers. The results showed that the Levenberg−Marquardt algorithm is a method of choice regarding both the fast convergence and the prediction of data not used in the training set. However, exceptions were observed when the number of weights was large as well as with the large number of data points.

In summary, the results show that the Levenberg−Marquardt method was superior over the conjugate gradient and delta rule methods as regarding convergence and prediction. This conclusion appears to be invariant with respect to the number of the training compounds and the number of input descriptors in a typical QSAR/QSPR neural network architecture. Therefore, the results from this work suggest of using preferably the Levenberg−Marquardt method as an optimizer in the nonlinear QSAR/QSPR modeling since it can "handle" most of the practical cases better.

ACKNOWLEDGMENT

We thank Dr. Sulev Sild and Dr. Andre Lomaka for their help in this project.

REFERENCES AND NOTES

(1) Hopfield, J. J. Neurons with Graded Response Have Collective Computational Properties Like Those with Two-State Neurons. *Proc. Nat. Acad. Sci.* **1984**, *81*, 3088.

(2) Jurs, P. C. Computer Applications in Chemistry: A University Course. *Comput. Educ. Chem.* (*Symp.*) **1984**, 1.
(3) Zupan, J.; Gasteiger, J. Neural Networks: A New Method for Solving Chemical Problems or Just a Passing Phase? *Anal. Chim. Acta* **1991**, *248*, 1
(4) Baskin, I. I.; Ait A. O.; Halberstam, N. M.; Palyulin, V. A.; Zefirov, N. S. An Approach to the Interpretation of Backpropagation Neural Network Models in QSAR Studies. SAR *QSAR Environ. Res*. **2002**, *13*, 35.
(5) Karelson, M.; Sild, S.; Maran, U. Nonlinear QSAR Treatment of Genotoxicity. *Mol. Simulat.* **2000**, *24*, 229.
(6) Sild, S.; Karelson, M. A General Treatment for Dielectric Constants of Organic Compounds. *J. Chem. Inf. Comput. Sci*. **2002**, *42*, 360.
(7) Katritzky, A. R.; Dobchev, D. A.; Fara, D. C.; Karelson, M. QSAR Studies on 1-phenylbenzimidazoles as Inhibitors of the Platelet-Derived Growth Factor. *Bioorg. Med. Chem.* **2005**, *13*, 6598.
(8) Rumelhart, D.; Hinton, G.; Williams, R. Learning Presentation by Backpropagation Errors. *Nature* **1986**, *323*, 533.
(9) Vogl, T.; Mangis, A.; Zigler, W. Accelerating the Convergence of the Backpropagation Method. *Bio. Cyber.* **1988**, *59*, 256.
(10) Barnard, E. Optimization for Training Neural Networks. *IEEE Trans. Neural Net.* **1992**, *3*, 232.
(11) Cheralabous, C. Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks. *IEE Proc.* **1992**, *139*, 301.
(12) Kollias, S.; Anastssiou, D. An adaptive least squares algorithm for the efficient training of ANN. *IEEE Trans. Circ. Syst*. **1989**, *36*, 1092.
(13) Hagan, M.; Menhaj, M. Training Networks with the Marquardt Algorithm. **1994**, *5*, 989.
(14) Tetko, I. V.; Tanchuk, V. Y. Application of Associative Neural Networks for Prediction of Lipophilicity in ALOGPS 2.1 Program. *J. Chem. Inf. Comput. Sci*. **2002**, *42*, 1136.
(15) Haykin, S. *Neural Networks. A comprehensive foundation*; Pearson, Ed.; 1999.
(16) Jang, J.-S.; Sun, C.-T.; Mizutani, E. *Neuro-Fuzzy and Soft Computing; A Computational Approach to Learning and Machine Intelligence*,; Robbins, T., Ed.; Prentice Hall, Inc.: 1997.
(17) Marqurdt, D. An Algorithm for Least Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Math.* **1963**, 431.
(18) Press: W.; Teukolsky, S.; Vetterling, W.; Flannery, B. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed.; Cambridge University Press: New York, 1992.
(19) Matsumoto, M.; Nishimura, T. A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *Trans. Mod. Comp. Sim*. **1998**, *8*, 3.
(20) Galbershtam, N.; Baskin, I.; Paljulin, V.; Zefirov, N. *Uspehi Himii* **2003**, *72*, 706.
(21) *Codessa Pro Software*; University of Florida, 2002; www.codessa-pro.com.
(22) Katritzky, A. R.; Mu, L.; Lobanov, V. S.; Karelson, M. Correlation of Boiling Points with Molecular Structure. 1. A Training Set of 289 Diverse Organics and a Test Set of 9 Simple Inorganics. *J. Phys. Chem.* **1996**, *100*, 10400.
(23) Katritzky, A. R.; Wang, Y.; Sild S.; Tamm, T.; Karelson, M. QSPR Studies on Vapor Pressure, Aqueous Solubility, and the Prediction of Water−Air Partition Coefficients. *J. Chem. Inf. Comput. Sci*. **1998**, *38*, 720.
(24) Katritzky, A. R.; Dobchev, D. A.; Hur, E.; Fara, D. C.; Karelson, M. QSAR Treatment of Drugs Transfer into Human Breast Milk. *Bioorg. Med. Chem.* **2005**, *13*, 1623.
(25) Hemmateenejad, B. Correlation Ranking Procedure for Factor Selection in PC-ANN modeling and application to ADMETox Evaluation. *Chemom. Intell. Lab. Syst.* **2005**, *75*, 237.
(26) Fatemi, M. H. Prediction of Ozone Degradation Rate Constant of Organic Compounds by Using Artificial Neural Networks. *Anal. Chim. Acta* **2006**, *556*, 355.
(27) Guha, R.; Stanton, D. T.; Jurs, P. C. Interpreting Computational Neural Network Quantitative Structure−Property Relationship Models: A Detailed Interpretation of the Weights and Biases. *J. Chem. Inf. Model*. **2005**, *45*, 1109.
(28) Wang, Y.-H.; Li, Y.; Yang, S.-L.; Yang, L. An In−Silico Approach for Screening Flavonids as P-glycoprotein Inhibitors Based on a Bayesian-regularized Neural Network. *J. Comput.-Aided Mol. Des*. **2005**, *19*, 137.
(29) Karelson, M. *Molecular Descriptors in QSAR/QSPR*; Wiley & Sons: New York, 2000.

CI0600206