ELSEVIER

Journal of Molecular Structure (Theochem) 622 (2003) 71–83

THEO CHEM

www.elsevier.com/locate/theochem

Artificial neural networks and genetic algorithms in QSAR

Stefan P. Niculescu*

479-5 Woodview Road, Burlington, Ont., Canada L7N 2Z9

Abstract

Artificial neural networks are presented from the perspective of their potential use as modeling tools in quantitative structure–activity relationships (QSAR) research. First, general merits and drawbacks of the neural network modeling approach are discussed, and the relationship between neural networks, statistics and expert systems is clarified. A separate section is devoted exclusively to the subject of validating neural networks models. Next, the review focuses on presenting the most commonly used artificial neural networks in QSAR: backpropagation neural networks, probabilistic neural networks, Bayesian regularized neural networks, and Kohonen SOM. For each of them, both merits and shortcomings are revealed, and references are made to publications presenting their QSAR applications. Another section is devoted to genetic algorithms, their merits and shortcomings, and their potential use for model variables dimensionality reduction in QSAR studies. The last section is devoted to software resources.

Keywords: Quantitative structure–activity relationships; Neural networks; Genetic algorithms

1. Introduction

The first papers on artificial neural networks emerged almost 60 years ago, predating the advent of computers. Researchers were, and continue to be, fascinated with the way the human brain organizes and processes information, and especially by its capability to learn and recognize nonlinear patterns and trends, and adapt to new situations. For that reason, they tried to implement such capabilities in artificial computational devices using mathematical algorithms, commonly referred to as artificial neural networks. Genetic algorithms are search paradigms targeting combinatorial optimization, and are based on the Darwinian evolution principles. Modeling the relationships between chemical structure and various biological effects or activity of interest is the target of structure–activity studies. As the knowledge in the field of quantitative structure–activity relationships (QSAR) accumulates, it confirms the very high complexity and nonlinear character of most of such relationships. Artificial neural networks are among the best available tools to generate such nonlinear models, and genetic algorithms may be used to identify the fittest model according to various criteria. All neural networks we refer to are artificial neural networks. The rapidly increasing number of successful QSAR studies based on neural networks and genetic algorithms is the best proof of a very productive cooperation between Artificial Intelligence and Chemistry.

2. Artificial neural networks

Artificial neural networks are parallel computational devices consisting of groups of highly interconnected processing elements called neurons.

* Tel.: +1-905-634-4403.
E-mail address: stenic@globalserve.net (S.P. Niculescu).

0166-1280/03/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0166-1280(02)00619-X

Neural networks are characterized by topology, computational characteristics of their elements, and training rules. Traditional neural networks have neurons arranged in a series of layers. The first layer is termed the input layer, and each of its neurons receives information from the exterior, corresponding to one of the independent variables used as inputs. The last layer is the output layer, and its neurons handle the output from the network. The layers of neurons between the input and output layers are called hidden layers. Each layer may make its independent computations and may pass the results yet to another layer. In feed-forward neural networks the connections among neurons are directed upwards, i.e. connections are not allowed among the neurons of the same layer or the preceding layer. Networks where neurons are connected to themselves, with neurons in the same layer or neurons from a preceding layer, are termed feedback or recurrent networks. At a very simplified level, artificial neural networks mimic the way a biological brain organizes, stores, and processes information. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons, and the same happens with artificial neural networks. Neural networks may learn using only one type of learning: supervised (both input and output information used for training) or unsupervised (only input information is available for training). For example, backpropagation and probabilistic neural networks use supervised learning, while most of the self-organizing neural networks (Kohonen's Learning Vector Quantizer (LVQ), the Boltzman Machine, ART1 Implementing Adaptive Resonance Theory, etc.) are based on unsupervised learning. For the moment, the best available monographs on neural networks and their applications are Refs. [1–4]. Good and easy to understand introductions on the subject are presented in Refs. [5–15]. Ref. [16] introduces the neural networks from the perspective of their potential use as modeling tools in chemistry and drug design.

3. General advantages and drawbacks of neural networks

A very important advantage of neural networks is that they are *adaptive*, i.e. they can take data and learn from it. This ability does not depend upon the prior knowledge of rules. There are no conditions put on the predicted variables, they can be Boolean (True/False), continuous values, one or more classes among n, and so forth. Moreover, with only a few exceptions, neural networks are essentially nonlinear, and they are capable of learning complex interactions among the input variables in a system even when they are difficult to find and describe. Consequently, neural networks can provide solutions for problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found. Another related strength is their capability of extracting essential process information from data. As new training data become available, the neural network can be updated to represent the process more accurately. From the psychological point of view, the idea of learning is far easier to understand and accept than the complexities of advanced mathematics or statistics.

Another extremely important advantage of neural networks is that they are capable of *generalization*, i.e. they can correctly process information that only broadly resembles the original training data. They are also fault tolerant by being capable of properly handling noisy or incomplete data.

Perhaps the most important advantage of all is that neural networks allow going directly from factual data to the models without any human subjective interference, i.e. without tainting the result with *oversimplification* or *preconceived* ideas.

A neural network model describes associations, not causes and, therefore, it does not explain its decisions. Many view this fact as a drawback. In general, multivariate regression and related methodologies do not explain their decisions either. There is one notable exception, the combination between multivariate linear regression and principal component analysis. When applied to congeneric classes of compounds, such combination of methodologies may be extremely useful in elucidating the mechanism of action and in identifying which structural features of the molecules are most responsible for generating the effects of interest.

Another subjective objection against the use of neural networks is connected with model transparency. Most opponents of neural networks are promoting the idea that neural networks are black

boxes. This may be true for the novice user, but it is very far from reality for the computer expert. As long as a trained neural network may be retrieved and used for data processing this means that it is implemented through computer code, and consequently, possible to translate it into the form of a system of mathematical equations. Some neural network simulators have special modules allowing retrieval of the C++ computer code implementing a trained neural network. See Ref. [17] for a real life example.

4. Validation of neural network models

Any attempt to build a predictive model based on artificial neural networks generates the need to investigate how appropriate the network's architecture and computational paradigm are for the task that it is supposed to handle. For neural networks based on supervised learning, the data are usually split into a *training* set, a *training validation* set, and a *test set*. The training set consists of a set of examples used *only* for learning, (i.e. to fit the weights of the network). The training validation set is a set of examples used to adjust the network parameters such as network architecture (for example number of hidden layers and neurons, or number of training cycles). The test set is a set of examples used *only* to assess the generalization performance of a trained neural network, (external valuation) . Various networks are trained by minimization of an appropriate error function defined with respect to the training set. The performance of the networks is then compared by evaluating the error function using the training validation set, and the network having the smallest error with respect to the training validation set is selected. Any data set that is used to choose the best of two or more networks using the same architecture an computational paradigm, is by definition, a validation set, and the error of the chosen network on the validation set is *optimistically biased*. This has serious implications in evaluating the performance of models involving a large number of input variables or which are the result of combinations of various methodologies. In such modeling exercises, aside from the usual training validation and the external validation (required), performing a *full cross-validation* experiment (aiming to properly evaluate how appropriate the network architecture and its general computational paradigm are for the task the network is designed to handle) is strongly recommended. For the purpose of full cross-validation, the training data are randomly split into a number of disjoint cross-validation subsets. One of each of these subsets is left out in turn, and the remaining complement of data are used to train a partial model, using the same architecture, choice of input and output variables, and learning paradigm. The samples in the left-out data are then used to perform predictions. At the end of this process, there are predictions for all data in the training set, made-up from the predictions originating from the resulting partial models. All partial models are then assessed against the same performance criteria and decisions are made based on the consistency of the assessment results. When all cross-validation subsets consist of only one data point each, the procedure is known as a *leave-one-out cross-validation*. For larger data sets, it is practical to choose larger cross-validation subsets, usually of approximately the same size. A common choice is 20% of all training data, and in that case the procedure is known as a *leave-20%-out cross-validation*. For example, in the case of a small training data set, the number of training cycles necessary for a back-propagation neural network with prescribed architecture to reach its optimum efficiency may be approximated through a leave-one-out cross-validation experiment. For backpropagation neural network using large training sets and large number of model variables, the decision of selecting the optimal architecture as number of hidden layers and number of neurons in each of them, must be based on full cross-validation experiments. The same applies to the optimal feature selection in hybrid models combining genetic algorithms or principal component analysis with neural networks or multivariate regressions. The principles behind this validation methodology are as objective as they can be, and may be applied to any kind of predictor.

5. Relationships between neural networks and statistics

Occasionally, the principles behind the paradigms of some artificial neural networks are similar or

sometimes identical to their statistical counterparts; some examples are as follows. The feed-forward networks with no hidden layer are implementations of generalized linear models, while the ones with one hidden layer are related closely to projection pursuit regression. PNNs are based on the kernel discriminant analysis. Kohonen neural networks for adaptive vector quantization are similar to k-means cluster analysis, while the Kohonen self-organizing maps are discrete approximations to principal curves and surfaces. Principal component analysis and the Hebbian learning are strongly connected. Methods that are commonly used for fitting statistical nonlinear models, such as various Levenberg–Marquardt and conjugate gradient algorithms, can be used to train feed-forward networks. Statistics focus essentially on data analysis, while the neural networks do not. Consequently, when building neural network models, the data need to satisfy only *relaxed* distributional assumptions, if any. In contract, in order to ensure validity of the models generated by most statistical methodologies, very specific distributional assumptions on the model variables and data are required. See Refs. [3,4] for a detailed discussion of the relationship between statistics and neural networks. For comparisons of traditional and neural network classification models we recommend Refs. [18,19].

The most common purpose of QSAR modeling exercises is to build (mapping) models for biological activity prediction. Let us compare from this perspective the nonlinear multivariate regression with neural networks. Suppose the relationship subject to modeling has the general form $y = g(x_1, \ldots, x_k)$ where y is the target model variable, x_1 to x_k are the model variables, and g is an *unknown* multivariate nonlinear function. Given a (large enough) set of learning cases, the problem is to estimate g. At this level of generality only neural networks may approach the problem. Nonlinear multivariate regression may be used only when assumptions on the distribution of model errors are specified and the relationship is of the form $y = g(x_1, \ldots, x_k; \alpha_1, \ldots, \alpha_p)$ where g is a *known* function and only parameters α_1 to α_p are subject to estimation. The degree of nonlinearity and interactions of model variables are heavily influencing the particular choice of methodology used to find the appropriate solution. Presence of co-linearities in the data may also generate problems in applying traditional statistics. In contrast, neural networks do not require specification of the relationship, and there is no need to select the most important independent variables in the data set, as neural networks can automatically select them. The weights given by the network to the connections corresponding to irrelevant variables are negligible, while the ones corresponding to relevant variables are significant. The learning capability allows neural networks to detect complex and subtle interactions between the independent variables, contributing to the development of a model with increased accuracy. Robustness to presence of (moderate) noise in the data is another quality separating the mapping neural networks from traditional nonlinear multivariate regression, which is extremely sensitive to the presence of outliers. Regression models are static in the sense that the nature of the model cannot be changed, i.e. cannot learn and become smarter at a later time. The traditional regression model usually ignores the fact that (in general) data tend to be clustered around certain regions and skewed with respect to certain variables [20]. Significant underlying patterns may fail to be recognized by that approach. In contrast, artificial neural networks have the capability to automatically identify patterns between independent and dependent variables in a data set. In addition, they possess the ability to specify and estimate a specialized submodel or model adjustment for each pattern, and they can trigger and use such submodels appropriately as patterns reoccur. Feed-forward neural networks are better at learning moderately pathological relationships than are many other methods with stronger smoothness assumptions, as long as the number of pathological features (such as discontinuities) in the associated function is not too large. Of course, preliminary statistical analysis of raw data may help the development of a neural network. Hybrid statistical-neural networks systems can provide very useful solutions to some specific problems. It is necessary to remember that the objective of the modeling exercise is to create a model with significant predictive power. It is not enough to determine which relationships are statistically significant. Continuing this common theme, the practice reveals that the predictive power of traditional statistical models is considerably inferior to that of models based on neural networks. This does not mean that it is not possible (occasion-

ally) to obtain a superior mapping model using nonlinear multivariate regression [21]. The choice of which model is preferable must be decided on the basis of proper model validation and also what is required from the model.

6. Comparison between neural networks and expert systems

Expert systems are computer systems that simulate both judgment and behavior of a human or group that has expert knowledge in a given field. They achieve this by drawing inferences from a knowledge base containing accumulated experience and sets of rules for applying this knowledge to each particular situation presented to it. Inside an expert system the knowledge is represented as combinations of if-then rules and decision trees. An expert system can document its decisions according to its rules. In contrast, neural networks do not use any rules and model only associations, not causes. Expert systems have no tolerance for missing and erroneous values, while neural networks are fault tolerant and able to properly handle moderately noisy data. Expert systems cannot handle new cases for which rules are not available in their knowledge base. Each time the expertise is changed, the knowledge base must be (manually) updated. The new rules have to be specially formatted by a knowledge engineer starting from the human expert's knowledge. Neural networks may accommodate a new class of cases by simply updating their learning. Expert systems are very difficult and costly to build and maintain. In the case of neural networks, these operations are simple and inexpensive.

7. Neural networks in QSAR

There are various approaches researchers are using to build structure–activity models. Until very recently, most models targeting such relationships were (almost exclusively) based on variations of the classical multivariate regression paradigm and principal component analysis. Due to the restrictions imposed by the mathematical assumptions behind the statistical methodology, the validity of these models was limited to very narrow classes of compounds only, usually consisting of small structural variations. Advances in chemistry, biology and medicine revealed that the relationship between structure and effects is essentially nonlinear and highly complex, practically out of the reach of classical statistical modeling techniques. Frustrated modelers were forced to look elsewhere for solutions. The application of neural networks to most of these problems may provide better and more comprehensive results. The purpose for which models are built is strongly influencing the selection of model parameters. For instance, environmentalists will be interested in using such models to screen large lists of high volume industrial chemicals for toxic potential towards various biological species of interest. This kind of model must accommodate very diverse chemical structures, so they must account for the presence and contribution of a very large number of molecular fragments involved in various modes of toxic action. On the other side of the spectrum, modelers interested in the design of more efficient drugs will consider more restrictive classes of compounds, sharing for instance the same molecular backbone. The variables involved in such models must reflect the specific mode of action or binding to a receptor molecule, for example. In these cases, the parameters must account for the substituents' shape, position, local surface area charge, etc. While the neural networks may help in identifying models, the genetic algorithms may help in selecting the right combination of input parameters. The first applications of neural networks in QSAR modeling emerged approximately 12 years ago in the field of pharmacology and drug design [22–28]. Step by step, neural networks are recognized as valuable tools in investigating QSARs and QSPRs. Next we will describe only the most used neural networks paradigms, genetic algorithms and mixed models, and we discuss their merits and shortcomings. For QSARs related to Hopfield neural networks, Boltzmann machines, adaptive resonance theory (ART) models, and inductive logic programming see Refs. [29–34].

8. Backpropagation neural networks in QSAR: merits and shortcomings

Backpropagation neural networks (BNNs) are multilayer feed-forward neural networks trained by

backpropagation of errors (traditionally) using *gradient descent* or *conjugate gradient* algorithms [2]. As a consequence of the celebrated Stone–Weierstrass theorem, all three-layer (one hidden) feed-forward neural networks whose neurons use arbitrary squashing activation functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy [35]. So, in practical terms, the BNN is a universal approximator, and (at least in theory) it is possible to teach it anything that can be learned. This explains its great popularity with the QSAR modelers. Once identified, BNNs are very fast and easy to use. However, training a BNN is a very delicate task as it is slow and there is no guarantee that the achieved minimum is global. A first challenge in training the BNN is the choice of the appropriate network architecture, i.e. number of hidden layers and number of neurons each layer. There is no available rigorous theoretical result on which such choice may rely on. A practical solution is to start with a three-layer architecture where the number of neurons in the hidden layer is the geometric mean of the numbers of neurons in the input and output layers (geometric pyramid rule), and then experiment with it by adding and removing neurons in the hidden layer. For a discussion on the effects of changing the number of neurons in the hidden layer on the network performance see Refs. [36,37]. The speed of the training/learning is strongly affected by the learning rates and momentum parameters involved in training the connections between consecutive network layers.

Commonly, in most computer implementations of the backpropagation algorithm, the learning rates are all equal, and the same is forced on the momentum parameters. Too small values for the learning rates will slow down the convergence of the training process, while too large values will make the errors explode. The momentum parameters must be activated only when the convergence of the training process is too slow. Their role is to force the connection weights to change and speed up the convergence by adding a portion of the previous change in weights. Appropriate values for the learning rates and momentum parameters are identified through trial and error. See Ref. [38] for a discussion on the effects of changing the learning rates. The learning process is initialized using random seeding of the connection weights. This has serious implications on the resulting model reproducibility, although workarounds are available [39]. The BNN training involves repeated passes through the training data, and with each pass the network learns more. In turn, this creates another challenge, namely how to identify the right number of cycles necessary for the network to learn without loosing generality. Proper validation of the training, according to the neural networks validation principles discussed before is needed. The information in the network is stored through the trained connections. Consequently, the size of the training set must be large enough to allow the learning process to converge. Caution is strongly recommended in using the BNN paradigm in the case of small training data sets. The methodology requires at least one and a half as many training cases as connections in the network. Most computer programs implementing BNN simulators accept data only in a prescribed range. Consequently, data preprocessing is necessary. A great number of QSAR modelers favor the use of so called min/max interval data preprocessing. This consists of the linear transforms of the [min, max] ranges of input/output values of the individuals in the training set into the range required by the simulator (usually [0,1] or $[-1,1]$). In this case, the neural network is only able to handle those members of the external test set whose input values are within the same [min, max] range. To avoid such an event, some modelers prefer to select first the external test set, based on various subjective criteria. Next they isolate a representative training validation subset from the remaining data, while using the leftovers as training set. This is a recipe for generating overfitted models, which always have limited generalization capabilities. Better choices of preprocessing policies, overcoming the limitations of the min/max transforms, consist of combinations of Z-transforms, sigmoid logistic or hyperbolic tangent functions, and linear compression into the range required by the simulator. With such transforms, the training, training validation, and the external test sets may be selected using random selection, which represents the most objective choice. Of course, performing a full cross-validation experiment (based on random selection) will generate a better idea of the model's capabilities. While BNNs are universal approximators, the modeler must be aware that they are capable to handle only

a limited number of discontinuities. To properly validate models where a large proportion of input variables have a Boolean character (see for example Ref. [40]), full cross-validation based on random selection coupled with external validation must be performed. It should be noted that there is no guarantee that the achieved minimum, resulting from the BNN training based on gradient, is global. This problem may be overcome by using modified learning algorithms based on simulated annealing or genetic optimization [2]. Examples of BNN QSAR models targeting biodegradation and persistency are presented in Refs. [41–46]. Diverse BNN QSAR models for the acute toxicity of chemicals to various species of interest may be found in Refs. [17,47–54]. Models for carcinogenicity and mutagenicity are investigated in Refs. [55–57]. In vitro 5-lipoxygenase inhibitory potency and estrogen binding affinities are targeted in Refs. [58,59]. Examples of BNN QSAR models for the activity of various classes of drugs are presented in Refs. [60–66]). ReNDeR (reversible nonlinear dimension reduction) neural network is a special case of five-layers BNN targeting model data dimensionality reduction [67–69]. Using BNNs to model structure–odor relationships is the focus of Refs. [70–73]. For an example of BNN model for the prediction of secondary structure of proteins see Ref. [16].

9. Probabilistic neural networks in QSAR: merits and shortcomings

The PNN is in essence a combination of neural networks and Bayesian statistics and implements a practical solution (based on Parzen kernels, and spheres of influence) for the following mathematical problem: how to approximate the unknown distribution of a given population based on a learning set consisting of multivariate sample data, and without making any assumptions on the nature of the distribution itself. For theoretical background and details of the algorithm we refer to Refs. [2,74,75]. Once the estimator is built, the predictions are generated via the well-known Bayes' Theorem. The most common choice of kernel is the basic Gaussian kernel, which involves only the Gaussian function and only one sphere of influence parameter σ. As Bayesian approximators, PNNs may be used for both mapping (value prediction) and classification. According to their purpose, the architecture of the PNNs follows very precise rules. For instance, the basic Gaussian kernel PNN built to map x_k as function of $x_1, x_2, \ldots, x_{k-1}$ will have $k-1$ neurons in the input layer, one neuron in the first hidden layer for each case in the training set, $k-1$ neurons in the second hidden layer (summation layer), and one neuron in the output layer. The activation functions are de facto conditional statistical estimators associated with the training population, and the only parameter really subject to training (using Meisel [76] rules) is σ. This explains why the learning phase of the PNN involves only one pass through the training data, and there is no need for a training validation set. Judging a PNN's generalization performance is handled through external validation. Validating the model using full cross-validation experiments based on random selection is advised. The PNN's training is so fast that for the case of a small number of input and output variables it can be performed in real-time. In addition, the PNN is very robust, learning only the essential information from outliers, and being able to handle sparse samples. The basic Gaussian kernel mapping PNN's performance is comparable (usually slightly inferior) to that of the best BNNs built to handle the same task, but without the BNN's aggravations. A major drawback of the PNN is that as the number of training cases grows, the same happens with the memory requirements associated with the second network layer and its connections. The methodology requires at least as many training cases as input variables in the network. It is recommended to use it only for small up to moderately large training sets (a few thousand training cases). Large size of the training set impacts negatively on the computational speed of the resulting PNN. The most used data preprocessing policies consist of combinations of Z-transforms, sigmoid logistic or hyperbolic tangent functions, and linear compression into the range required by the simulator. Use of the principal components methodology to perform orthogonalization of the model input variables is recommended. Selecting the appropriate kernel is a very delicate and complex experimental process. There is a strong connection between the kernel choice and the PNN's learning and generalization performance. For instance, it is very common

that the PNN with reciprocal kernel generates overfitted models, recognizing suprisingly well the training data, but performing poorly outside. Proper validation of any PNN model is a must. Only recently the PNN started to be used for QSAR purposes. The first applications were in aquatic toxicology and targeted the toxicity of chemicals to various species of interest [77–81]. Other models investigate the relative binding affinity of steroids to the progesterone receptor [82] and of the carboxylic acid esters and organochlorine compounds to the estrogen receptor [83].

10. Bayesian regularized neural networks and QSAR: merits and shortcomings

Bayesian regularized artificial neural networks (BRANNs) are multilayer feed-forward neural networks obtained through Bayesian training [84]. Conventional training results in a single set of weights that can be used by the network to generate predictions. In contrast, the Bayesian training produces a posterior distribution over network weights. When the inputs of the network are set to the values for some new case, the posterior distribution over network weights will give rise to a corresponding predictive distribution over the outputs. If a single-valued prediction is needed, then the mean of the predictive distribution may be used as network output, but the full predictive distribution also provides information on how uncertain this prediction is. Choosing the appropriate network architecture (number of hidden layers and number of hidden units in each layer) and adapting to the relevant characteristics of the data depends on using the right prior distribution. Usually, such prior distributions are defined in a hierarchical fashion, using hyperparameters. One of the most common approaches, based on the Levenberg–Marquart algorithm with regularization, is to find first the weights and hyperparameters that are most probable, and then to approximate the distribution over weights. Special automatic relevance detection methods were developed, allowing all input parameters to be used in the neural net, with the Bayesian inference eliminating those containing no information or redundant information. A more general approach based on Markov chain Monte Carlo simulation is also available [85]. The Bayesian training allows the use of all training data for building the model. Nevertheless, proof of the model's generalization power may be confirmed only through external validation. The methodology requires at least twice as many training cases as weights in the network, and training is slow. BRANNs are robust, well matched to the data, and revealing good generalization capabilities. However, statements claiming that the resulting models are the best possible are greatly exaggerated. BRANN type QSARs which model the binding activity of two classes of drugs to the $GABA_A$ and M_1 muscarinic receptors are presented in Ref. [86], a similar model for the acute toxicity of substituted benzenes towards *Tetrahymena pyriformis* is discussed in Ref. [87], and recognition of the 'drug-like' character of compounds using BRANNs is investigated in Ref. [88]. Related models targeting the rodent carcinogenicity of chemicals are investigated in Ref. [89]. See Ref. [90] as well.

11. Kohonen neural networks and QSAR: merits and shortcomings

Kohonen's self-organizing map (SOM) is targeting clustering, visualization, and abstraction [91]. The basic concept behind the SOMs is *preservation of topology* (relationship among data). A SOM is a one active layer neural network consisting of a multidimensional array of neurons (usually two-dimensional). Each neuron in the grid is also an output neuron. The neurons are connected only with their closest neighbors in the array according to a prescribed topological scheme. The local feedback has the result that topologically close neurons react similarly when they receive similar input. In other words, if a particular neuron represents a given pattern, then its neighbors represent similar patterns. The SOM is trained through unsupervised competitive learning using a 'winner takes it all' policy. All neurons in the active layer obtain the same multidimensional input, and at the same time. A training case is presented to the network, and the winning neuron

found. That winner has its weights updated using the current learning rate, while the learning rate for the neighbors is scaled down proportional to the distance to the winner. Consequently, the knowledge of that pattern will be localized in the area of the winner. Any number of inputs may be used as long as the number of inputs is greater than the dimensionality of the grid space. Each training cycle involves one pass through the data and the training is stopped when changes to the network's weights become insignificant. A new case is classified to the cluster associated with the corresponding winner neuron of the grid. The Kohonen's LVQ is simply a supervised learning version of a SOM where inputs and expected output categories are presented to the network for training. The SOM expects data in a given range. The first step in data preprocessing is to insure that all variables match the network's domain. The second is to normalize the data, such that the final data vector is of unit length. Such restrictions are needed to ensure convergence of the training process. Both SOM and LVQ are relatively weak classifiers as they are strictly linear in their response. They train relatively fast, and are easy to interpret. Various SOM applications to clustering protein sequences may be found in Refs. [91–95]. Molecular similarity and molecular diversity are targeted in Refs. [96–98]. A SOM-based template approach for the comparison of the shape and molecular electrostatic potential values on the van der Waals surface of a molecule with that of a reference structure is investigated in Ref. [99]. See Refs. [16,100] for more examples involving the molecular electrostatic potential. Visual exploration of QSAR data based on SOMs is discussed in Ref. [101]. See Ref. [102] for an example of LVQ neural network for classification of substances according to their toxic mode of action. Use of SOMs for selection of test series is investigated in Refs. [91,101]. SOM layers may be combined with other neural network type layers. For instance, the counter-propagation neural network is the result of attaching a Grossberg layer (using Hebbian learning) to a SOM. Examples of counter-propagation neural network QSARs may be found in Refs. [103,104]. A hybrid virtual screening system for identifying olfactory compounds using a combination of SOM and fuzzy clustering is presented in Ref. [105].

12. Genetic algorithms and QSAR: merits and shortcomings

Genetic algorithms are implementations of various search paradigms inspired by natural evolution. At a very general level, a genetic algorithm may be any (chromosome-type) population-based model that uses selection and recombination operators to generate new sample points in a search space. Definition of the search space and the search goal are problem-dependent. For most QSAR modelers the search space is usually a space of solutions for a specified problem (trained neural networks, multiple nonlinear regression models, etc.), and the goal is to identify that solution which performs the best according to an evaluation function, such as, for example, the average square error generated by a model on a test set. Suppose the problem is of feature-selection type, where each solution is implementing a relationship using a particular selection of input parameters. Each input parameter is uniquely associated with a chromosome gene. The first step is to choose the size of the chromosomes and to put in place an encoding scheme uniquely mapping combinations of model parameters of the same size with chromosomes. The next step is to generate the initial population of chromosomes. Implementing genetic competition requires definition of a fitness function for the chromosome population. For example, in the case of the classical genetic algorithm, the fitness of a chromosome is represented as the ratio of the value of the evaluation function for the model corresponding to that chromosome over the average value generated by applying the evaluation function to the models associated with all chromosomes. The genetic algorithm involves a two-stage process. It starts with the current population. Selection (using stochastic sampling with replacement or remainder stochastic sampling) is applied to the current population to create an intermediate population. The next population is the result of genetic manipulation of the chromosomes in the intermediate population through recombination (crossover) and/or mutation. This process of going from the current population to

the next population represents one generation execution of a genetic algorithm. Crossover occurs when there is an exchange of genes between chromosomes. Mutation consists of replacing (randomly) some of the chromosome's genes with genes not present in the chromosome and its role is to restore lost genetic material. The chromosomes are evaluated after each cycle using the fitness function. Generation of new populations is repeated until a satisfactory solution is identified, or specific termination criteria are met. For a general review on the genetic algorithms and their applications in chemistry and chemometrics I recommend Refs. [106,107]. Applications in QSAR and drug design are targeted in Ref. [108]. Use of genetic algorithms is advised when it is impossible to apply an exhaustive search, for instance when the response hypersurface in which the optimum is searched is of high dimension and has many local optima. The presence of local optima makes a direct optimization based on the steepest ascent unreliable, since it can be caught in a local optimum far away from the global one. The genetic algorithms generally have good search accuracy, i.e. they approach the globally optimal solution irrespective of the starting condition, but a poor search precision, i.e. they commonly fail in finding the optimum solution. In contrast, the traditional local optimization techniques have a good search precision and poor search accuracy. If there exists a good specialized optimization method for a specific problem, then the genetic algorithm may not be the best optimization tool to use. The main use of genetic algorithms in QSAR targets variable selection and model identification [16,109,110]. The genetic algorithm handles the selection, while the model paradigm generates the evaluation function. Examples of QSARs modeling the steroids binding affinity to the progesterone receptor and obtained through combinations of genetic algorithms and BNNs are investigated in Refs. [111,112]. A combination of genetic algorithm with a feed-forward fully connected three-layer neural network trained by the BFGS quasi-Newton method (CNN) has been used to model the acute toxicity of chemicals to fish [113]. The lack of full cross-validation is a definite minus of such studies, and is caused by the researchers, not the methodology. Combinations of genetic algorithms and BNNs may also be used to solve the inverse problem of designing molecules with specific properties or activity [114,115]. An application of genetic algorithms in test series selection is discussed in Ref. [116]. For applications of the genetic function approximation algorithm in QSAR studies see Refs. [117–119]. Use of genetic algorithms for variable selection in modeling in vitro antifilarial activity of antimicyn A1 analogues is reported in Ref. [120]. The use of a model generated by a combination of genetic algorithms and BNN to the design of new high potency benzodiazepines was illustrated in Ref. [121].

13. Neural networks and genetic algorithms software

From the many available commercial software packages implementing a large variety of neural network paradigms and genetic algorithms *NeuroShell* and *STATISTICA: Neural Networks* stand out. NeuroShell is produced and distributed by *Ward Systems Group, Inc.* (Executive Park West, 5 Hillcrest Drive, Frederick, Maryland 21702, USA; http://www.wardsystems.com). It is a Microsoft Windows software package, which may be used to create a large variety of BNN, Kohonen SOM, PNN (classification) and GRNN models. With all options installed, NeuroShell allows the retrieval of trained neural networks in the form of C++ computer code that may be easily translated into mathematical equations fully describing a model or may be used to build software applications implementing it. STATISTICA: Neural Networks is produced and distributed by *StatSoft, Inc.* (2300 E. 14th St., Tulsa, OK 74104, USA; http://www.statsoft.com). It is a Microsoft Windows application capable of creating a large variety of neural network models (BNN, Kohonen SOM, PNN (classification), GRNN, etc.) and offering a number of unique features such as genetic input selection and automatic network design. Both companies offer comprehensive training for the potential users, and both software packages include a large number of case studies and easy to use manuals. Unix platform software for flexible Bayesian modeling and Markov chain sampling is available as freeware via http://www.cs.toronto.edu/~radford/fbm.software.html. For the scientists interested in designing their own software implementing neural network type

QSAR models, *Neuro Windows* and the *Genetic Server/Library* may be very useful. Neuro Windows, produced and distributed by *Ward Systems Group, Inc.*, is a powerful library of neural networks functions. It is targeting BNN, PNN, Kohonen (SOM) and GRNN algorithms. The Genetic Server/ Library is produced and distributed by *NeuroDimension Inc.* (1800 N. Main Street, Suite D4, Gainesville, FL 32609, USA; http:/www.nd.com), and provides general-purpose functions for genetic algorithm design and implementation. Both products are compatible with the Microsoft Visual Developer Studio.

14. Conclusions

This review of the major types of neural networks and related methodologies and their applications to QSAR studies clearly demonstrates their potential to advance the field. While some specific methods may be more appropriate than others for a particular problem at hand, overall, their general ability to model nonlinear effects makes them useful tools to deal with the sets of noncongeneric structures, unknown or varying modes of actions, and other reasons for the failure of traditional linear models in many instances. The rapidly increasing volume of QSAR studies based on neural networks is a strong indication of their usefulness and success. Further scientific advances within this field and their widespread acceptance and use are expected to follow.

Acknowledgements

I am thankful to Dr K.L.E. Kaiser and Environment Canada for supporting part of this work. I express my gratitude to the references for their constructive comments.

References

[1] J. Hertz, A. Krogh, R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA, 1991.
[2] T.A. Masters, Practical Neural Network Recipes in C++, Academic Press, San Diego, CA, 1993.
[3] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[4] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.
[5] I. Aleksander, H. Morton, An Introduction to Neural Computing, Chapman & Hall, London, 1990.
[6] R. Beale, T. Jackson, Neural Computing, An Introduction, Adam Hilger, IOP Publishing Ltd, Bristol, 1990.
[7] J.E. Dayhoff, Neural Network Architectures: An Introduction, Van Nostrand Reinhold, New York, 1990.
[8] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley, Reading, MA, 1990.
[9] B. Muller, J. Reinhardt, Neural Networks, An Introduction, Springer, Berlin, 1990.
[10] G.A. Orchard, W.A. Phillips, Neural Computation: A Beginner's Guide, Lawrence Earlbaum Associates, London, 1991.
[11] M.J. Zurada, Itroduction to Artificial Neural Systems, West Publishing Company, St. Paul, MN, 1992.
[12] V.B. Rao, H.V. Rao, C++ Neural Networks and Fuzzy Logic, MIS Press, New York, 1993.
[13] P.D. Wasserman, Advanced Methods in Neural Computing, Van Nostrand Reinhold, New York, 1993.
[14] L.V. Fausett, Fundamentals of Neural Networks: Architectures, Algorithms and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1994.
[15] S. Haykin, Neural Networks, A Comprehensive Foundation, Macmillan, New York, 1994.
[16] J. Zupan, J. Gasteiger, Neural Networks in Chemistry and Drug Design, second ed., Wiley-VCH, Weinheim, 1999.
[17] K.L.E. Kaiser, S.P. Niculescu, G. Schüürmann, Water Qual. Res. J. Can. 32 (1997) 637.
[18] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, 1994.
[19] P.V. Balakrishnan, M.C. Cooper, V.S. Jacob, P.A. Lewis, Psychometrika 59 (1994) 509.
[20] B. Joseph, F.H. Wang, S.S. Shieh, Comput. Chem. Engng 16 (1992) 413.
[21] B. Lucic, N. Trinajstic, J. Chem. Inf. Comput. Sci. 39 (1999) 121.
[22] T. Aoyama, Y. Suzuki, H. Ichikawa, J. Med. Chem. 33 (1990) 905.
[23] T. Aoyama, Y. Suzuki, H. Ichikawa, J. Med. Chem. 33 (1990) 2583.
[24] T.A. Andrea, H. Kalayeh, in: C. Silipo, A. Vittoria (Eds.), QSAR: Rational Approaches to the Design of Bioactive Compounds, Elsevier, Amsterdam, 1991, p. 209.
[25] T. Aoyama, H. Ichikawa, Chem. Pharm. Bull. 39 (1991) 358.
[26] T. Aoyama, H. Ichikawa, Chem. Pharm. Bull. 39 (1991) 372.
[27] J.Y. de Saint Laumer, M. Chastrette, J. Devillers, in: J. Devillers, W. Karcher (Eds.), Applied Multivariate Analysis in SAR and Environmental Studies, Kluwer, Dordrecht, 1991, p. 479.
[28] J.D. Hirst, M.J.E. Stemberg, Biochemistry 31 (1992) 7211.
[29] J.P. Doucet, A. Panaye, SAR QSAR Environ. Res. 8 (1998) 249.
[30] F. Gagné, C. Blaise, Chemosphere 35 (1997) 1343.

[31] D. Domine, J. Devillers, D. Wienke, L. Buydens, J. Chem. Inf. Comput. Sci. 37 (1997) 10.
[32] D. Wienke, D. Domine, L. Buydens, J. Devillers, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 119.
[33] J.D. Hirst, R.D. King, M.J.E. Stemberg, J. Comput. Aided Mol. Des. 8 (1994) 405.
[34] J.D. Hirst, R.D. King, M.J.E. Stemberg, J. Comput. Aided Mol. Des. 8 (1994) 421.
[35] K. Homik, M. Stinchcombe, H. White, Neural Networks 2 (1989) 359.
[36] D.T. Manallack, D.J. Livingstone, Med. Chem. Res. 2 (1992) 181.
[37] D.J. Livingstone, D.T. Manallack, J. Med. Chem. 36 (1993) 1295.
[38] G. Schüürmann, E. Mtiller, Environ. Toxicol. Chem. 13 (1994) 743.
[39] I.V. Tetko, D.J. Livingstone, A.I. Luik, J. Chem. Inf. Comput. Sci. 35 (1995) 826.
[40] J. Devillers, Toxicol. Meth. 10 (2000) 69.
[41] V. Zitko, Chemosphere 23 (1991) 305.
[42] B. Chambon, J. Devillers, Quant. Struct.–Act. Relat. 12 (1993) 49.
[43] J. Devillers, SAR QSAR Environ. Res. 1 (1993) 161.
[44] D. Domine, J. Devillers, M. Chastrette, W. Karcher, SAR QSAR Environ. Res. 1 (1993) 211.
[45] J. Devillers, D. Domine, R.S. Boethling, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 65.
[46] E. Rorijie, M.C. van Wezel, W.J.G.M. Peijnenburg, SAR QSAR Environ. Res. 4 (1995) 219.
[47] L. Xu, J.W. Ball, S.L. Dixon, P.C. Jurs, Environ. Toxicol. Chem. 13 (1994) 841.
[48] J. Devillers, S. Bintein, D. Domine, W. Karcher, SAR QSAR Environ. Res. 4 (1995) 29.
[49] D. Zakarya, E.M. Larfoui, A. Boulaamail, T. Lakhlifi, SAR QSAR Environ. Res. 5 (1996) 269.
[50] K.L.E. Kaiser, S.P. Niculescu, G. Schüürmann, Water Qual. Res. J. Can. 32 (1997) 637 see also page 855.
[51] D. Zakarya, A. Boulaamail, E.M. Larfoui, T. Lakhlifi, SAR QSAR Environ. Res. 6 (1997) 181.
[52] J. Devillers, D. Domine, SAR QSAR Environ. Res. 10 (1999) 61.
[53] J. Devillers, J. Flatin, SAR QSAR Environ. Res. 11 (2000) 25.
[54] J. Devillers, SAR QSAR Environ. Res. 11 (2001) 397.
[55] D. Villemin, D. Cherqaoui, A. Mesbah, J. Chem. Inf. Comput. Sci. 34 (1994) 1288.
[56] M. Brinn, M. Payne, P.T. Walsh, Chem. Engng Res. Des. 71-A3 (1993) 337.
[57] M. Brinn, P. Walsh, M.P. Payne, B. Bott, SAR QSAR Environ. Res. 1 (1993) 169.
[58] N. Ghoshal, S.N. Mulkhopadhyay, T.K. Ghoshal, B. Achari, Bioorg. Med. Chem. Lett. 3 (1993) 329.
[59] N. Ghoshal, S.N. Mulkbopadhayay, T.K. Ghoshal, B. Achari, Indian J. Chem. 32B (1993) 1045.
[60] T.A. Andrea, H. Kalayeh, J. Med. Chem. 34 (1991) 2824.
[61] T. Suzuki, M. Ishida, Pharm. Sci. 1 (6) (1995) 297.
[62] Y. Tang, H.-W. Wang, K.-X. Chen, R.-Y. Ji, Zhongguo Yaoli Xuebao 16 (1995) 26.
[63] Z. Li, F. Hu, B. Liang, Y.H. Benxi, L. Shi, M. Li, M. Sasaki, Yaoxue Xuebao 31 (1996) 38.
[64] D.J. Maddalena, Expert Opin. Ther. Pat. 6 (1996) 239.
[65] M. Hosseini, D.J. Maddalena, I. Spence, J. Chem. Inf. Comput. Sci. 37 (1997) 1129.
[66] D.J. Maddalena, G.A.R. Johnston, J. Med. Chem. 38 (1995) 715.
[67] D.J. Livingstone, G. Hasketh, D. Clayworth, J. Mol. Graph. 9 (1991) 115.
[68] D.J. Livingstone, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 157.
[69] D.T. Manallack, T. Gallager, D.J. Livingstone, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 177.
[70] M. Chastrette, J.Y. de Saint Laumer, Eur. J. Med. Chem. 26 (1991) 829.
[71] M. Chastrett, J.Y. de Saint Laumer, J.F. Peyraud, SAR QSAR Environ. Res. 1 (1993) 221.
[72] M. Chastrett, C. El Aidi, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 83.
[73] J. Devillers, C. Guillon, D. Domine, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug esign, Academic Press, London, 1996, p. 97.
[74] D. Specht, Proc. IEEE Int. Conf. Neural Networks 1 (1988) 525.
[75] D. Specht, Neural Networks 3 (1990) 109.
[76] W.S. Meisel, Computer-oriented Approaches to Pattern Recognition, Academic Press, New York, 1972.
[77] S.P. Niculescu, K.L.E. Kaiser, G. Schüürmann, Water Qual. Res. J. Can. 33 (1998) 153.
[78] K.L.E. Kaiser, S.P. Niculescu, Chemosphere 38 (1999) 3237.
[79] S.P. Niculescu, K.L.E. Kaiser, T.W. Schultz, Arch. Environ. Toxicol. Chem. 39 (2000) 289.
[80] K.L.E. Kaiser, S.P. Niculescu, Environ. Toxicol. Chem. 20 (2001) 420.
[81] K.L.E. Kaiser, S.P. Niculescu, T.W. Schultz, SAR QSAR Environ. Res. 13 (2002) 57.
[82] S.P. Niculescu, K.L.E. Kaiser, Quant. Struct.–Act. Relat. 20 (2001) 223.
[83] K.L.E. Kaiser, S.P. Niculescu, Water Qual. Res. J. Can. 36 (2001) 619.
[84] D.J.C. Mackay, Comput. Neural Syst. 6 (1995) 469.
[85] R.M. Neal, Bayesian Learning for Neural Networks, Springer, New York, 1996.
[86] F.R. Burden, D.A. Winkler, J. Med. Chem. 42 (1999) 3183.
[87] F.R. Burden, D.A. Winkler, Chem. Res. Toxicol. 13 (2000) 436.
[88] A. Ajay, W.P. Walters, M. Murcko, J. Med. Chem. 41 (1998) 3314.
[89] D. Bahler, B. Stone, C. Wellington, D.W. Bristol, J. Chem. Inf. Comput. Sci. 40 (2000) 906.
[90] F.R. Burden, D.A. Winkler, Mol. Simul. 24 (2000) 243.

[91] T. Kohonen, Self-organizing Maps, Springer, Berlin, 1995.
[92] P. Arrigo, F. Giuliano, F. Scalia, A. Rapallo, G. Damiani, Comput. Appl. Biosci. 7 (1991) 353.
[93] E.A. Ferran, P. Ferrara, Biol. Cybern. 65 (1991) 451.
[94] E.A. Ferran, P. Ferrara, Comput. Appl. Biosci. 8 (1992) 39.
[95] E.A. Ferran, B. Pflugfelder, Comput. Appl. Biosci. 9 (1993) 671.
[96] T.W. Barlow, J. Mol. Graph. 13 (1995) 24.
[97] P. Bernard, A. Golbraikh, D. Kireev, J.R. Chretien, N. Rozhkova, Analysis 26 (1998) 333.
[98] D. Kireev, J.R. Chrétien, P. Bernard, F. Ros, SAR QSAR Environ. Res. 8 (1998) 93.
[99] S. Anzali, G. Barnickel, M. Krug, J. Sadowski, M. Wagner, J. Gasteiger, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 209.
[100] M. Wagener, J. Sadowski, J. Gasteiger, J. Am. Chem. Soc. 117 (1995) 7769.
[101] D. Domine, D. Wienke, J. Devillers, L. Buydens, in: J. Devillers (Ed.), Neural Networks in QSAR and Drug Design, Academic Press, London, 1996, p. 223.
[102] S.C. Basak, G.D. Grunwald, G.E. Host, G.J. Niemi, S.P. Bradbury, Environ. Toxicol. Chem. 17 (1998) 1056.
[103] K.L. Peterson, Anal. Chem. 64 (1992) 379.
[104] K.L. Peterson, J. Chem. Inf. Comput. Sci. 35 (1995) 896.
[105] F. Ros, K. Audouze, M. Pintore, J.R. Chrétien, SAR QSAR Environ. Res. 11 (2000) 281.
[106] R. Leardi, J. Chemometrics 15 (2001) 559.
[107] B.T. Luke, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 35.
[108] J. Devillers, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 1.
[109] H. Kubinyi, Quant. Struct.–Act. Relat. 13 (1994) 393.
[110] R. Leardi, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 67.
[111] S.P. van Helden, H. Hamersma, V.J. van Geerestein, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 159.
[112] S.-S. So, S.P. van Helden, V.J. van Geerestein, M. Karplus, J. Chem. Inf. Comput. Sci. 40 (2000) 762.
[113] D.V. Eldred, C.L. Weikel, P.C. Jurs, K.L.E. Kaiser, Chem. Res. Toxicol. 12 (1999) 670.
[114] J. Devillers, C. Putavy, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 303.
[115] V. Venkatasubramanian, A. Sundaram, K. Chan, J.M. Caruthers, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 271.
[116] C. Putavy, J. Devillers, D. Domine, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 143.
[117] D. Rogers, A.J. Hopfinger, J. Chem. Inf. Comput. Sci. 34 (1994) 854.
[118] D. Rogers, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 87.
[119] W.J. Dunn, D. Rogers, in: J. Devillers (Ed.), Genetic Algorithms in Molecular Modeling, Academic Press, London, 1996, p. 109.
[120] S.-S. So, M. Karplus, J. Med. Chem. 39 (1996) 1521.
[121] S.-S. So, M. Karplus, J. Med. Chem. 39 (1996) 5246.