ELSEVIER

Chemometrics and Intelligent Laboratory Systems 39 (1997) 43–62

Chemometrics and intelligent laboratory systems

Tutorial

Introduction to multi-layer feed-forward neural networks

Daniel Svozil [a,*], Vladimír Kvasnička [b], Jiří Pospíchal [b]

[a] *Department of Analytical Chemistry, Faculty of Science, Charles University, Albertov 2030, Prague, CZ-12840, Czech Republic*
[b] *Department of Mathematics, Faculty of Chemical Technology, Slovak Technical University, Bratislava, SK-81237, Slovakia*

Received 15 October 1996; revised 25 February 1997; accepted 6 June 1997

Abstract

Basic definitions concerning the multi-layer feed-forward neural networks are given. The back-propagation training algorithm is explained. Partial derivatives of the objective function with respect to the weight and threshold coefficients are derived. These derivatives are valuable for an adaptation process of the considered neural network. Training and generalisation of multi-layer feed-forward neural networks are discussed. Improvements of the standard back-propagation algorithm are reviewed. Example of the use of multi-layer feed-forward neural networks for prediction of carbon-13 NMR chemical shifts of alkanes is given. Further applications of neural networks in chemistry are reviewed. Advantages and disadvantages of multilayer feed-forward neural networks are discussed. © 1997 Elsevier Science B.V.

Keywords: Neural networks; Back-propagation network

Contents

[*] Corresponding author.

PII S0169-7439(97)00061-0

1. Introduction

Artificial neural networks (ANNs) [1] are networks of simple processing elements (called 'neurons') operating on their local data and communicating with other elements. The design of ANNs was motivated by the structure of a real brain, but the processing elements and the architectures used in ANN have gone far from their biological inspiration.

There exist many types of neural networks, e.g. see [2], but the basic principles are very similar. Each neuron in the network is able to receive input signals, to process them and to send an output signal. Each neuron is connected at least with one neuron, and each connection is evaluated by a real number, called the weight coefficient, that reflects the degree of importance of the given connection in the neural network.

In principle, neural network has the power of a universal approximator, i.e. it can realise an arbitrary mapping of one vector space onto another vector space [3]. The main advantage of neural networks is the fact, that they are able to use some a priori unknown information hidden in data (but they are not able to extract it). Process of 'capturing' the unknown information is called 'learning of neural network' or 'training of neural network'. In mathematical formalism to learn means to adjust the weight coefficients in such a way that some conditions are fulfilled.

There exist two main types of training process: supervised and unsupervised training. Supervised training (e.g. multi-layer feed-forward (MLF) neural network) means, that neural network knows the desired output and adjusting of weight coefficients is done in such way, that the calculated and desired outputs are as close as possible. Unsupervised training (e.g. Kohonen network [4]) means, that the desired output is not known, the system is provided with a group of facts (patterns) and then left to itself to settle down (or not) to a stable state in some number of iterations.

2. Multi-layer feed-forward (MLF) neural networks

MLF neural networks, trained with a back-propagation learning algorithm, are the most popular neural networks. They are applied to a wide variety of chemistry related problems [5].

A MLF neural network consists of neurons, that are ordered into layers (Fig. 1). The first layer is called the input layer, the last layer is called the out-

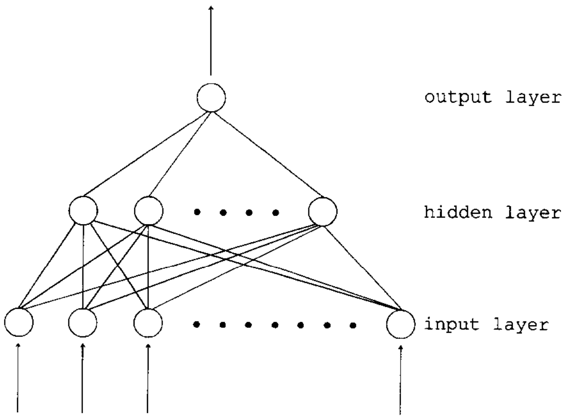

Fig. 1. Typical feed-forward neural network composed of three layers.

put layer, and the layers between are hidden layers. For the formal description of the neurons we can use the so-called mapping function Γ, that assigns for each neuron i a subset $\Gamma(i) \subseteq V$ which consists of all ancestors of the given neuron. A subset $\Gamma^{-1}(i) \subseteq V$ than consists of all predecessors of the given neuron i. Each neuron in a particular layer is connected with all neurons in the next layer. The connection between the ith and jth neuron is characterised by the weight coefficient ω_{ij} and the ith neuron by the threshold coefficient ϑ_i (Fig. 2). The weight coefficient reflects the degree of importance of the given connection in the neural network. The output value (activity) of the ith neuron x_i is determined by Eqs. (1) and (2)). It holds that:

$$x_i = f(\xi_i) \tag{1}$$

$$\xi_i = \vartheta_i + \sum_{j \in \Gamma_i^{-1}} \omega_{ij} x_j \tag{2}$$

where ξ_i is the potential of the ith neuron and function $f(\xi_i)$ is the so-called transfer function (the summation in Eq. (2) is carried out over all neurons j transferring the signal to the ith neuron). The threshold coefficient can be understood as a weight coefficient of the connection with formally added neuron j, where $x_j = 1$ (so-called bias).

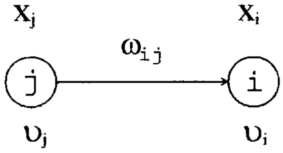

Fig. 2. Connection between two neurons i and j.

For the transfer function it holds that

$$f(\xi) = \frac{1}{1 + \exp(-\xi)} \tag{3}$$

The supervised adaptation process varies the threshold coefficients ϑ_i and weight coefficients ω_{ij} to minimise the sum of the squared differences between the computed and required output values. This is accomplished by minimisation of the objective function E:

$$E = \sum_o \tfrac{1}{2}(\boldsymbol{x}_o - \hat{\boldsymbol{x}}_o)^2 \tag{4}$$

where $\boldsymbol{x}_o$ and $\hat{\boldsymbol{x}}_o$ are vectors composed of the computed and required activities of the output neurons and summation runs over all output neurons o.

3. Back-propagation training algorithm

In back-propagation algorithm the steepest-descent minimisation method is used. For adjustment of the weight and threshold coefficients it holds that:

$$\begin{aligned} \omega_{ij}^{(k+1)} &= \omega_{ij}^{(k)} - \lambda \left(\frac{\partial E}{\partial \omega_{ij}} \right)^{(k)} \\ \vartheta_i^{(k+1)} &= \vartheta_i^{(k)} - \lambda \left(\frac{\partial E}{\partial \vartheta_i} \right)^{(k)} \end{aligned} \tag{5}$$

where λ is the rate of learning ($\lambda > 0$). The key problem is calculation of the derivatives $\partial E / \partial \omega_{ij}$ a $\partial E / \partial \vartheta_i$. Calculation goes through next steps:

First step

$$E = \tfrac{1}{2}(\boldsymbol{x}_o - \hat{\boldsymbol{x}}_o)^2 = \tfrac{1}{2} \sum_k g_k^2 \tag{6}$$

where $g_k = x_k - \hat{x}_k$ for $k \in$ output layer, $g_k = 0$ for $k \notin$ output layer

Second step

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial \omega_{ij}} = \frac{\partial E}{\partial x_i}\frac{\partial f(\xi_i)}{\partial \omega_{ij}}$$

$$= \frac{\partial E}{\partial x_i}\frac{\partial f(\xi_i)}{\partial \xi_i}\frac{\partial \xi_i}{\partial \omega_{ij}}$$

$$= \frac{\partial E}{\partial x_i} f'(\xi_i)\frac{\partial(\sum_{j\in \Gamma_i^{-1}} \omega_{ij} x_j + \vartheta_i)}{\partial \omega_{ij}}$$

$$= \frac{\partial E}{\partial x_i} f'(\xi_i) x_j \qquad (7)$$

$$\frac{\partial E}{\partial \vartheta_i} = \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial \vartheta_{ij}} = \frac{\partial E}{\partial x_i} f'(\xi_i)\cdot l \qquad (8)$$

From Eqs. (7) and (8) results the following important relationship

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial \vartheta_i}\cdot x_j \qquad (9)$$

Third step

For the next computations is enough to calculate only $\partial E/\partial \vartheta_i$.

$i \in$ output layer

$$\frac{\partial E}{\partial x_i} = g_i \qquad (10)$$

$i \in$ hidden layer

$$\frac{\partial E}{\partial x_i} = \sum_{l\in \Gamma_i}\frac{\partial E}{\partial x_l}\frac{\partial x_l}{\partial x_i} = \sum_{l\in \Gamma_i}\frac{\partial E}{\partial x_l}\frac{\partial f(\xi_l)}{\partial x_i}$$

$$= \sum_{l\in \Gamma_i}\frac{\partial E}{\partial x_l}\frac{\partial f(\xi_l)}{\partial \xi_l}\frac{\partial \xi_l}{\partial x_i} = \sum_{l\in \Gamma_i}\frac{\partial E}{\partial x_l} f'(\xi_l)\,\omega_{li}$$

$$= \sum_{l\in \Gamma_i}\frac{\partial E}{\partial \vartheta_l}\omega_{li} \qquad (11)$$

because

$$\frac{\partial E}{\partial x_l} f'(\xi_l) = \frac{\partial E}{\partial \vartheta_l},$$

(see Eq. (8))

Based on the above given approach the derivatives of the objective function for the output layer and then for the hidden layers can be recurrently calculated. This algorithm is called the back-propagation, because the output error propagates from the output layer through the hidden layers to the input layer.

4. Training and generalisation

The MLF neural network operates in two modes: training and prediction mode. For the training of the MLF neural network and for the prediction using the MLF neural network we need two data sets, the training set and the set that we want to predict (test set).

The training mode begins with arbitrary values of the weights – they might be random numbers – and proceeds iteratively. Each iteration of the complete training set is called an epoch. In each epoch the network adjusts the weights in the direction that reduces the error (see back-propagation algorithm). As the iterative process of incremental adjustment continues, the weights gradually converge to the locally optimal set of values. Many epochs are usually required before training is completed.

For a given training set, back-propagation learning may proceed in one of two basic ways: pattern mode and batch mode. In the pattern mode of back-propagation learning, weight updating is performed after the presentation of each training pattern. In the batch mode of back-propagation learning, weight updating is performed after the presentation of all the training examples (i.e. after the whole epoch). From an 'on-line' point of view, the pattern mode is preferred over the batch mode, because it requires less local storage for each synaptic connection. Moreover, given that the patterns are presented to the network in a random manner, the use of pattern-by-pattern updating of weights makes the search in weight space stochastic, which makes it less likely for the back-propagation algorithm to be trapped in a local minimum. On the other hand, the use of batch mode of training provides a more accurate estimate of the gradient vector. Pattern mode is necessary to use for example in on-line process control, because there are not all of training patterns available in the given time. In the final analysis the relative effectiveness of the two training modes depends on the solved problem [6,7].

In prediction mode, information flows forward through the network, from inputs to outputs. The net-

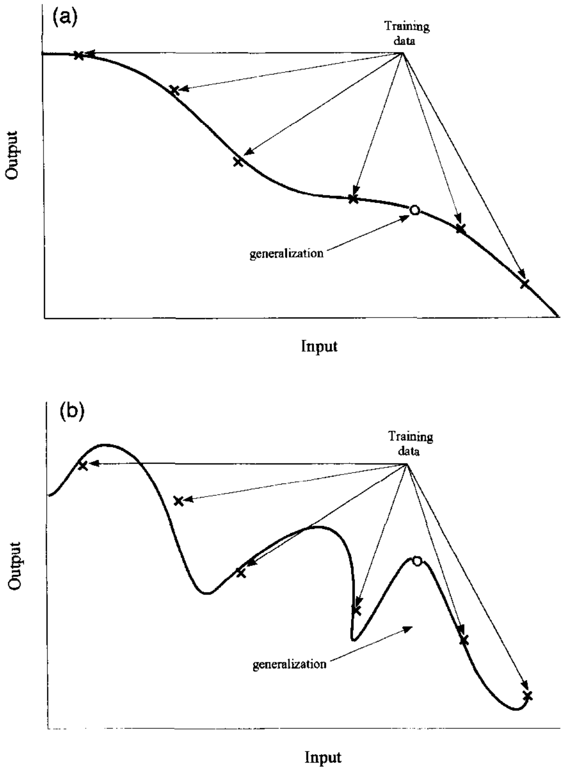

Fig. 3. Principle of generalisation and overfitting. (a) Properly fitted data (good generalisation). (b) Overfitted data (poor generalisation).

work processes one example at a time, producing an estimate of the output value(s) based on the input values. The resulting error is used as an estimate of the quality of prediction of the trained network.

In back-propagation learning, we usually start with a training set and use the back-propagation algorithm to compute the synaptic weights of the network. The hope is that the neural network so designed will generalise. A network is said to generalise well when the input-output relationship computed by network is correct (or nearly correct) for input/output patterns never used in training the network. Generalisation is not a mystical property of neural networks, but it can be compared to the effect of a good non-linear interpolation of the input data [8]. Principle of generalisation is shown in Fig. 3a. When the learning process is repeated too many iterations (i.e. the neural network is overtrained or overfitted, between overtrainig and overfitting is no difference), the network may memorise the training data and therefore be less able to generalise between similar input–output patterns. The network gives nearly perfect results for examples from the training set, but fails for examples from the test set. Overfitting can be compared to improper choose of the degree of polynom in the polynomial regression (Fig. 3b). Severe overfitting can occur with noisy data, even when there are many more training cases than weights.

The basic condition for good generalisation is sufficiently large set of the training cases. This training set must be in the same time representative subset of the set of all cases that you want to generalise to. The importance of this condition is related to the fact that there are two different types of generalisation: interpolation and extrapolation. Interpolation applies to cases that are more or less surrounded by nearby training cases; everything else is extrapolation. In particular, cases that are outside the range of the training data require extrapolation. Interpolation can often be done reliably, but extrapolation is notoriously unreliable. Hence it is important to have sufficient training data to avoid the need for extrapolation. Methods for selecting good training sets arise from experimental design [9].

For an elementary discussion of overfitting, see [10]. For a more rigorous approach, see the article by Geman et al. [11].

Given a fixed amount of training data, there are some effective approaches to avoiding overfitting, and hence getting good generalisation:

4.1. Model selection

The crucial question in the model selection is 'How many hidden units should I use?'. Some books and articles offer 'rules of thumb' for choosing a topology, for example the size of the hidden layer to be somewhere between the input layer size and the output layer size [12] [1], or some other rules, but such rules are total nonsense. There is no way to determine a good network topology just from the number of inputs and outputs. It depends critically on the number of training cases, the amount of noise, and the

[1] Warning: this book is really bad.

complexity of the function or classification you are trying to learn. An intelligent choice of the number of hidden units depends on whether you are using early stopping (see later) or some other form of regularisation (see weight decay). If not, you must simply try many networks with different numbers of hidden units, estimate the generalisation error for each one, and choose the network with the minimum estimated generalisation error.

Other problem in model selection is how many hidden layers use. In multi-layer feed forward neural network with any of continuous non-linear hidden-layer activation functions, one hidden layer with an arbitrarily large number of units suffices for the 'universal approximation' property [13–15]. Anyway, there is no theoretical reason to use more than two hidden layers. In [16] was given a constructive proof about the limits (large, but limits nonetheless) on the number of hidden neurons in two-hidden neural networks. In practise, we need two hidden layers for the learning of the function, that is mostly continuous, but has a few discontinuities [17]. Unfortunately, using two hidden layers exacerbates the problem of local minima, and it is important to use lots of random initialisations or other methods for global optimisation. Other problem is, that the additional hidden layer makes the gradient more unstable, i.e. that training process slows dramatically. It is strongly recommended use one hidden layer and then, if using a large number of hidden neurons does not solve the problem, it may be worth trying the second hidden layer.

4.2. Weight decay

Weight decay adds a penalty term to the error function. The usual penalty is the sum of squared weights times a decay constant. In a linear model, this form of weight decay is equivalent to ridge regression. Weight decay is a subset of regularisation methods. The penalty term in weight decay, by definition, penalises large weights. Other regularisation methods may involve not only the weights but various derivatives of the output function [15]. The weight decay penalty term causes the weights to converge to smaller absolute values than they otherwise would. Large weights can hurt generalisation in two different ways. Excessively large weights leading to hidden units can cause the output function to be too rough, possibly with near discontinuities. Excessively large weights leading to output units can cause wild outputs far beyond the range of the data if the output activation function is not bounded to the same range as the data. The main risk with large weights is that the non-linear node outputs could be in one of the flat parts of the transfer function, where the derivative is zero. In such case the learning is irreversibily stoped. This is why Fahlman [41] proposed to use the modification $f(\xi)(1-f(\xi))+0.1$ instead of $f(\xi)(1-f(\xi))$ (see p. 17). The offset term allows the continuation of the learning even with large weights. To put it another way, large weights can cause excessive variance of the output [11]. For discussion of weight decay see for example [18].

4.3. Early stopping

Early stopping is the most commonly used method for avoiding overfitting. The principle of early stopping is to divide data into two sets, training and validation, and compute the validation error periodically during training. Training is stopped when the validation error rate starts to go up. It is important to realise that the validation error is not a good estimate of the generalisation error. One method for getting an estimate of the generalisation error is to run the net on a third set of data, the test set, that is not used at all during the training process [19]. The disadvantage of split-sample validation is that it reduces the amount of data available for both training and validation.

Other possibility how to get an estimate of the generalisation is to use the so-called cross-validation [20]. Cross-validation is an improvement on split-sample validation that allows you to use all of the data for training. In k-fold cross-validation, you divide the data into k subsets of equal size. You train the net k times, each time leaving out one of the subsets from training, but using only the omitted subset to compute whatever error criterion interests you. If k equals the sample size, this is called leave-one-out cross-validation. While various people have suggested that cross-validation be applied to early stopping, the proper way of doing that is not obvious. The disadvantage of cross-validation is that you have to retrain the net many times. But in the case of MLF neural networks the variability between the results obtained on different trials is often caused with the fact, that the learning was ended up in many different local minima. Therefore the cross-validation method is more suitable for neural networks without the danger to fall into local minima (e.g. radial basis function, RBF, neural networks [83]). There exist a method similar to the cross-validation, the so-called bootstrapping [21,22]. Bootstrapping seems to work better than cross-validation in many cases.

Early stopping has its advantages (it is fast, it requires only one major decision by the user: what proportion of validation cases to use) but also some disadvantages (how many patterns are used for training and for validation set [23], how to split data into training and test set, how to know that validation error really goes up).

5. Advantages and disadvantages of MLF neural networks

The application of MLF neural networks offers the following useful properties and capabilities:

(1) *Learning*. ANNs are able to adapt without assistance of the user.

(2) *Nonlinearity*. A neuron is a non-linear device. Consequently, a neural network is itself non-linear. Nonlinearity is very important property, particularly, if the relationship between input and output is inherently non-linear.

(3) *Input–output mapping*. In supervised training, each example consists of a unique input signal and the corresponding desired response. An example picked from the training set is presented to the network, and the weight coefficients are modified so as to minimise the difference between the desired output and the actual response of the network. The training of the network is repeated for many examples in the training set until the network reaches the stable state. Thus the network learns from the examples by constructing an input–output mapping for the problem.

(4) *Robustness*. MLF neural networks are very robust, i.e. their performance degrades gracefully in the presence of increasing amounts of noise (contrary e.g. to PLS).

However, there are some problems and disadvantages of ANNs too. For some problems approximation via sigmoidal functions ANNs are slowly converging – a reflection of the fact that no physical insight is used in the construction of the approximating mapping of parameters on the result. The big problem is the fact, that ANNs cannot explain their prediction, the processes taking place during the training of a network are not well interpretable and this area is still under development [24,25]. The number of weights in an ANN is usually quite large and time for training the ANN is too high.

6. Improvements of back-propagation algorithm

The main difficulty of standard back-propagation algorithm, as it was described earlier, is its slow convergence, which is a typical problem for simple gradient descent methods. As a result, a large number of modifications based on heuristic arguments have been proposed to improve the performance of standard back-propagation. From the point of view of optimisation theory, the difference between the desired output and the actual output of an MLF neural network produces an error value which can be expressed as a function of the network weights. Training the network becomes an optimisation problem to minimise the error function, which may also be considered an objective or cost function. There are two possibilities to modify convergence behaviour, first to modify the objective function and second to modify the procedure by which the objective function is optimised. In a MLF neural network, the units (and therefore the weights) can be distinguished by their connectivity, for example whether they are in the output or the hidden layer. This gives rise to a third family of possible modifications, differential scaling.

6.1. Modifications to the objective function and differential scaling

Differential scaling strategies and modifications to the objective function of standard back-propagation are usually suggested by heuristic arguments. Modifications to the objective function include the use of different error metrics and output or transfer functions.

Several logarithmic metrics have been proposed as an alternative to the quadratic error of standard back-propagation. For a speech recognition problem, Franzini [26] reported a reduction of 50% in learning time using

$$E = -\sum_{p}\sum_{o}\ln\left(1-(x_o-\hat{x}_o)^2\right) \tag{12}$$

compared to quadratic error (p is the number of patterns, o is the number of output neurons). The most frequently used alternative error metrics are motivated by information theoretic learning paradigms [27,28]. A commonly used form, often referred to as the cross-entropy function, is

$$E = \sum_{k}\left[-\hat{x}_o\cdot\ln(x_o)-(1-\hat{x}_o)\cdot\ln(1-x_o)\right] \tag{13}$$

Training a network to minimise the cross-entropy objective function can be interpreted as minimising the Kullback–Liebler information distance [29] or maximising the mutual information [30]. Faster learning has frequently been reported for information theoretic error metrics compared to the quadratic error [31,32]. Learning with logarithmic error metrics was also less prone to get stuck in a local minima [31,32].

The sigmoid logistic function used by standard back-propagation algorithm can be generalised to

$$f(\xi) = \frac{K}{1+\exp(-D\cdot\xi)} - L \tag{14}$$

In standard back-propagation $K = D = 1$ and $L = 0$. The parameter D (sharpness or slope) of the sigmoidal transfer function can be absorbed into weights without loss of generality [33] and it is therefore set to one in most treatments. Lee and Bien [34] found that a network was able to more closely approximate a complex non-linear function when the back-propagation algorithm included learning the parameters K, D and L as well as weights. A bipolar sigmoid function (tanh) with asymptotic bounds at -1 and $+1$ is frequently used to increase the convergence speed. Other considerations have led to the use of different functions [35] or approximations [36].

Scaling the learning rate of a unit by its connectivity leads to units in different layers having different values of learning rate. The simplest version, dividing learning rate by the fan-in (the fan-in of a unit is the number of input connections it has with units in the preceding layer), is frequently used [37,38]. Other scaling methods with higher order dependence to fan-in or involving the number of connections between a layer and both its preceding and succeeding layers have also been proposed to improve convergence [39,40]. Samad [36] replaced the derivative of the logistic function $f'(\xi) = f(\xi)(1-f(\xi))$ for the output unit by its maximum value of 0.25 as well as dividing the backpropagated error by the fan-out (the fan-out of the unit is the number of output connections it has to units in the succeeding layer) of the source unit. Fahlman [41] found that $f(\xi)(1-f(\xi)) + 0.1$ worked better than either $f(\xi)(1-f(\xi))$ or its total removal from the error formulae.

6.2. Modifications to the optimisation algorithm

Optimisation procedures can be broadly classified into zero-order methods (more often referred to as minimisation without evaluating derivatives) which make use of function evaluations only, first order methods which make additional use of the gradient vector (first partial derivatives) and second order methods that make additional use of the Hessian (matrix of second partial derivatives) or its inverse. In general, higher order methods converge in fewer iterations and more accurately than lower order methods because of the extra information they employ but they require more computation per iteration.

Minimisation using only function evaluation is a little problematic, because these methods do not scale well to problems having in excess of about 100 parameters (weights). However Battiti and Tecchiolli [42] employed two variants of the adaptive random search algorithm (usually referred as random walk [43]) and reported similar results both in speed and generalisation to back-propagation with adaptive stepsize. The strategy in random walk is to fix a step size and attempt to take a step in any random direction from the current position. If the error decreases, the step is taken or else another direction is tried. If after a certain number of attempts a step cannot be taken, the stepsize is reduced and another round of attempts is tried. The algorithm terminates when a step cannot be taken without reducing the stepsize below a threshold value. The main disadvantage of random walk is that its success depends upon a careful choice of many tuning parameters. Another algorithm using only function evaluations is the polytope, in which the network weights form the vertices of a polytope [44]. The polytope algorithm is slow but is able to reduce the result of objective function to a lower value than standard back-propagation [45]. In the last years also some stochastic minimisation algorithms, as e.g. simulated annealing [46,47], were tried for adjusting the weight coefficients [48]. The disadvantage of these algorithms is their slowness, if their parameters are set so, that algorithms should converge into global minima of the objective function. With faster learning they tend to fall into deep narrow local minima, with results similar to overfitting. In practice they are therefore usually let run for a short time, and the resulting weights are used as initial parameters for backpropagation.

Classical steepest descent algorithm without the momentum is reported [42] to be very slow to converge because it oscillates from side to side across the ravine. The addition of a momentum term can help overcome this problem because the step direction is no longer steepest descent but modified by the previous direction.

$$\begin{aligned} \omega_{ij}^{(k+1)} &= \omega_{ij}^{(k)} - \lambda \left(\frac{\partial E}{\partial \omega_{ij}} \right)^{(k)} + \alpha \Delta \omega_{ij}^{(k)} \\ \vartheta_i^{(k+1)} &= \vartheta_i^{(k)} - \lambda \left(\frac{\partial E}{\partial \vartheta_i} \right)^{(k)} + \alpha \Delta \vartheta_i^{(k)} \end{aligned} \tag{15}$$

where α is the mometum factor ($\alpha \in \langle 0, 1 \rangle$). In effect, momentum utilises second order information but requires only one step memory and uses only local information. In order to overcome the poor convergence properties of standard back-propagation, numerous attempts to adapt learning rate and momentum have been reported. Vogl et al. [49] adapted both learning step and momentum according to the change in error on the last step or iteration. Another adaptive strategy is to modify the learning parameters according to changes in step direction as opposed to changes in the error value. A measure of the change in step direction is gradient correlation or the angle between the gradient vectors ∇E_n and ∇E_{n-1}. The learning rules have several versions [26,50]. Like standard back-propagation the above adaptive algorithms have one value of learning term for each weight in the network. Another option is to have an adaptive learning rate for each weight in the network. Jacobs [51] proposed four heuristics to achieve faster rates of convergence. A more parsimonious strategy, called SuperSAB [52], learned three times faster than standard back-propagation. Other two methods that are effective are Quickprop [43] and RPROP [53]. Chen and Mars [54] report an adaptive strategy which can be implemented in pattern mode learning and which incorporates the value of the error change between iterations directly into the scaling of learning rate.

Newton's method for optimisation uses Hessian matrix of second partial derivatives to compute step length and direction. For small scale problems where the second derivatives are easily calculated the method is extremely efficient but it does not scale well to larger problems because not only the second partial derivatives have to be calculated at each iteration but the Hessian must also be inverted. A way how to avoid this problem is to compute an approximation to the Hessian or its inverse iteratively. Such methods are described as quasi-Newton or variable metric. There are two frequently used versions of quasi-Newton: the Davidson–Fletcher–Powell (DFP) algorithm and the Broydon–Fletcher–Goldfarb–Shanno (BFGS) algorithm. In practise, van der Smagt [55] found DFP to converge to a minimum in only one third of 10000 trials. In a comparison study, Barnard [56] found the BFGS algorithm to be similar in average performance to conjugate gradient. In a function estimation problem [45], BFGS was able to reduce the error to a lower value than conjugate gradient, standard back-propagation and a polytope algorithm without derivatives. Only the Levenberg–Marquardt method [57–59] reduced the error to a lower value than BFGS. The main disadvantage of these methods is that storage space of Hessian matrix is proportional to the squarednumber of weights of the network.

An alternative second-order minimisation technique is conjugate gradient optimisation [60–62]. This algorithm restricts each step direction to be conjugate to all previous step directions. This restriction simplifies the computation greatly because it is no longer necessary to store or calculate the Hessian or its inverse. There exist two main versions of conjugate gradients: Fletcher–Reeves version [63] and Polak–Ribiere version [64]. The later version is said to be faster and more accurate because the former makes more simplifying assumptions. Performance comparison of standard back-propagation and traditional conjugate gradients seems to be task dependent. For example, according to [55] Fletcher–Reeves conjugate gradients were not as good as standard back-propagation on the XOR task but better than standard back-propagation on two function estimation tasks. Another point of comparison between algorithms is their ability to reduce error on learning the training set. De Groot and Wurtz [45] report that conjugate gradients were able to reduce error on a function estimation problem some 1000 times than standard back-propagation in 10 s of CPU time. Comparing conjugate gradients and standard back-propagation without momentum on three different classification tasks, method of conjugate gradients was able to reduce the error more rapidly and to a lower value than back-propagation for the given number of iterations [65]. Since most of the computational burden in conjugate gradients algorithms involves the line search, it would be an advantage to avoid line searches by calculating the stepsize analytically. Moller [66] has introduced an algorithm, which did this, making use of gradient difference information.

7. Applications of neural networks in chemistry

Interests in applications of neural networks in chemistry have grown rapidly since 1986. The number of articles concerning applications of neural networks in chemistry has an exponentially increasing tendency ([5], p. 161). In this part some papers dealing with the use of back-propagation MLF neural networks in chemistry will be reviewed. Such papers cover a broad spectrum of tasks, e.g. theoretical aspects of use of the neural networks, various problems in spectroscopy including calibration, study of chemical sensors applications, QSAR studies, proteins folding, process control in chemical industry, etc.

7.1. Theoretical aspects of the use of back-propagation MLF neural networks

Some theoretical aspects of neural networks were discussed in chemical literature. Tendency of MLF ANN to 'memorise' data (i.e. the predictive ability of network is substantially lowered, if the number of neurons in hidden layer is increased – parabolic dependence) is discussed in [67]. The network described in this article was characterised by a parameter ρ, that is the ratio of the number of data points in a learning set to the number of connections (i.e., the number of ANN internal degrees of freedom). This parameter was analysed also in [68,69]). In several other articles some attention was devoted to analysis of the ANN training. The mean square error MSE is used as a criterion of network training.

$$\mathrm{MSE} = \frac{\Sigma_p (x_o - \hat{x}_o)^2}{(\text{\# of compds.} \times \text{\# of out units})} \tag{16}$$

While the MSE for a learning set decreases with time of learning, predictive ability of the network has parabolic dependence. It is optimal to stop net training before complete convergence has occurred (the so-called 'early stopping') [70]. In [71] were shown benefits of statistical averaging of network prognosis. The problem of overfitting and the importance of cross-validation were studied in [72]. Some methods of the design of training and test set (i.e. methods raised from experimental design) were discussed in [9]. Together with the design of training and test set stands in the forefront of interest also a problem which variables to use as input into the neural networks ('feature selection'). For the determining the best subset of a set containing n variables there exist several possibilities:

- A complete analysis of all subsets. This analysis is possible only for small number of descriptors. It was reported only for linear regression analysis, not for the neural networks.
- A heuristic stepwise regression analysis. This type of methods includes forward, backward and Efroymson's forward stepwise regression based on the value of the F-test. Such heuristic approaches are widely used in regression analysis [73]. Another possibility is to use a stepwise model selection based on the Akaike information criterion [74]. Similar approaches were also described as methods for feature selection for neural networks [75].
- A genetic algorithm, evolutionary programming. Such methods were not used for neural networks because of their high computational demands. Application of these techniques for linear regression analysis was reported [76–78].
- Direct estimations (pruning methods). These

techniques are most widely used by the ANN researchers. An evaluation of a variable by such methods is done by introducing a sensitivity term for variable. Selection of variables by such methods in QSAR studies was pioneered by Wikel and Dow [79]. Several pruning methods were used and compared in [80].

Some work was also done in the field of improvement of the standard back-propagation algorithm, e.g. by use of the conjugate gradient algorithm [81] or the Flashcard Algorithm [82], that is reported to be able to avoid local minima. Other possibility to avoid local minima is to use another neural network architecture. Among the most promising belongs the radial basis neural (RBF) neural network [83]. RBF and MLF ANN were compared in [84].

7.2. Spectroscopy

The problem of establishing correlation between different types of spectra (infrared, NMR, UV, VIS, etc.) and the chemical structure of the corresponding compound is so crucial, that the back-propagation neural networks approach was applied in many spectroscopic problems. The main two directions in the use of neural networks for spectroscopy related problems are the evaluation of the given spectrum and the simulation of the spectrum of the given compound. Almost all existing spectra have been used as inputs to the neural networks (i.e. evaluation): NMR spectra [85–88], mass spectra [89–93], infrared spectra [94,95,84,96–98], fluorescence [99] and X-ray fluorescence spectra [100–102], gamma ray spectra [103,104], Auger electron spectra [105], Raman spectra [106,107], Mössbauer spectra [108], plasma spectra [109], circular dichroism spectra [110,111]. Another type of neural networks application in spectroscopy is the prediction of the spectrum of the given compound (Raman: [112], NMR: [113–115], IR: [116]).

7.3. Process control

In process control almost all the data come from non-linear equations or from non-linear processes and are therefore very hard to model and predict. Process control was one of the first fields in chemistry to which the neural network approach was applied. The basic problems in the process control and their solution using neural networks are described in [117]. The main goal of such studies is to receive a network that is able to predict a potential fault before it occurs [118,119]. Another goal of neural networks application in process control is control of the process itself. In [120] a method for extracting information from spectroscopic data was presented and studied by computer simulations. Using a reaction with non-trivial mechanism as model, outcomes in form of spectra were generated, coded, and fed into a neural network. Through proper training the network was able to capture the information concerning the reaction hyperplane, and predict outcomes of the reaction depending on past history. Kaiming et al. in their article [121] used a neural network control strategy for fed-batch baker's yeast cultivation. A non-linear single-input single-output system was identified by the neural network, where the input variable was the feed rate of glucose and the output variable was the ethanol concentration. The training of the neural network was done by using the data of on–off control. The explanation of results showed that such neural network could control the ethanol concentration at the setpoint effectively. In a review [122] are stated 27 references of approaches used to apply intelligent neural-like (i.e., neural network-type) signal processing procedures to solve a problem of acoustic emission and active ultrasonic process control measurement problems.

7.4. Protein folding

Proteins are made up of elementary building blocks, the amino acids. These amino acids are arranged sequentially in a protein, the sequence is called the primary structure. This linear structure folds and turns into three-dimensional structure that is referred as secondary structure (α-helix, β-sheet). Because the secondary structure of a protein is very important to biological activity of the protein, there is much interest in predicting the secondary structures of proteins from their primary structures. In recent years numerous papers have been published on the use of neural networks to predict secondary structure of proteins from their primary structure. The pioneers in this field were Qian and Sejnowski [123]. Since this date many neural networks systems for predicting secondary structure of proteins were developed. For example, Vieth et al. [124] developed a complex, cascaded neural network designed to predict the secondary structure of globular proteins. Usually the prediction of protein secondary structure by a neural network is based on three states (alpha-helix, beta-sheet and coil). However, there was a recent report of a protein with a more detailed secondary structure, the 310-helix. In application of a neural network to the prediction of multi-state secondary structures [125], some problems were discussed. The prediction of globular protein secondary structures was studied by a neural network. Application of a neural network with a modular architecture to the prediction of protein secondary structures (alpha-helix, beta-sheet and coil) was presented. Each module was a three-layer neural network. The results from the neural network with a modular architecture and with a simple three-layer structure were compared. The prediction accuracy by a neural network with a modular architecture was reported higher than the ordinary neural network. Some attempts were also done to predict tertiary structure of proteins. In [126] is described a software for the prediction of the 3-dimensional structure of protein backbones by neural network. This software was tested on the case of group of oxygen transport proteins. The success rate of the distance constraints reached 90%, which showed its reliability.

7.5. *Quantitative structure activity relationship*

Quantitative structure activity relationship (QSAR) or quantitative structure property relationship (QSPR) investigations in the past two decades have made significant progress in the search for quantitative relations between structure and property. The basic modelling method in these studies is a multilinear regression analysis. The non-linear relationships were successfully solved by neural networks, that in this case act as a function aproximator. The use of feed-forward back-propagation neural networks to perform the equivalence of multiple linear regression has been examined in [127] using artificial structured data sets and real literature data. Neural networks predictive ability has been assessed using leave-one-out cross-validation and training/test set protocols. While networks have been shown to fit data sets well, they appear to suffer from some disadvantages. In particular, they have performed poorly in prediction for the QSAR data examined in this work, they are susceptible to chance effects, and the relationships developed by the networks are difficult to interpret. Other comparison between multiple linear regression analysis and neural networks can be found in [128,129]. In a review (113 refs.) [130] QSAR analysis was found to be appropriate for use with food proteins. PLS (partial least-squares regression), neural networks, multiple regression analysis and PCR (principal component regression) were used for modelling of hydrophobity of food proteins and were compared. Neural networks can be also used to perform analytical computation of similarity of molecular electrostatic potential and molecular shape [131]. Concrete applications of the neural networks can be found for example in [132–135].

7.6. *Analytical chemistry*

The use of neural networks in analytical chemistry is not limited only to the field of spectroscopy. The general use of neural networks in analytical chemistry was discussed in [136]. Neural networks were successfully used for prediction of chromatography retention indices [137–139], or in analysis of chromatographic signals [140]. Also processing of signal from the chemical sensors was intensively studied [141–144].

8. Internet resources

In World-Wide-Web you can find many information resources concerning neural networks and their applications. This chapter will provide general information about such resources.

The news usenet group comp.ai.neural-nets is intended as a discussion forum about artificial neural networks. There is an archive of comp.ai.neural-nets on the WWW at http://asknpac.npac.syr.edu. The frequently asked question (FAQ) list from this newsgroup can be found in http://ftp://ftp.sas.com/pub/neural/FAQ.html. Others news groups partially connected with neural networks are comp.theory.self-org-sys, comp.ai.genetic and comp.ai.fuzzy.

The Internet mailing list dealing with all aspects of neural networks is called Neuron-Digest, to subscribe send e-mail to neuron-request@cattell.psych.upenn.edu.

Some articles about neural networks can be found in Journal of Artificial Intelligence Research, (http://www.cs.washington.edu/research/jair/home.html) or in Neural Edge Library (http://www.clients.globalweb.co.uk/nctt/newsletter/).

A very good and complex list of on-line and some off-line articles about all aspects of the back-propagation algorithm is the Backpropagator's review, (http://www.cs.washington.edu/research/jair/home.html).

The most complex set of technical reports, articles and Ph.D. thesis can be found at the so-called Neuroprose (ftp://archive.cis.ohio-state.edu/pub/neuroprose). Another large collection of neural network papers and software is at the Finish University Network (ftp://ftp.funet.fi/pub/sci/neural). It contains the major part of the public domain software and papers (e.g. mirror of Neuroprose). Many scientific groups dealing with neural network problems has their own WWW sites with downloadable technical reports, e.g. Electronic Circuit Design Workgroup (http://www.eeb.ele.tue.nl/neural/reports.html), Institute for Research in Cognitive Science (http://www.cis.upenn.edu/~ircs/Abstracts.html), UTCS (http://www.cs.utexas.edu/users/nn/pages/publications/publications.html), IDIAP (http://www.idiap.ch/html/idiap-networks.html) etc.

For the updated list of shareware/freeware neural network software look at http://www.emsl.pnl.gov:2080/docs/cie/neural/systems/shareware.html, for the list of commercial software look at StatSci (http://www.scitechint.com/neural.HTM) or at http://www.emsl.pnl.gov:2080/docs/cie/neural/systems/software.html. Very complex list of software is also available in FAQ. One of the best freeware neural network simulators is the Stuttgart Neural Network Simulator SNNS (http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html), that is targeted for Unix systems. MS-Windows front-end for SNNS (http://www.lans.ece.utexas.edu/winsnns.html) is available too.

For experimentation with neural networks there are available several databases, e.g. the neural-bench Benchmark collection (http://www.boltz.cs.cmu.edu/). For the full list see FAQ.

You can find nice list of NN societies in the WWW at http://www.emsl.pnl.gov:2080/docs/cie/neural/societies.html and at http://www.ieee.org:80/nnc/research/othernnsoc.html.

There is a WWW page for Announcements of Conferences, Workshops and Other Events on Neural Networks at IDIAP in Switzerland (http://www.idiap.ch/html/idiap-networks.html).

9. Example of the application – neural-network prediction of carbon-13 NMR chemical shifts of alkanes [2]

^{13}C NMR chemical shifts belong to the so-called local molecular properties, where it is possible to assign unambiguously the given property to an atom (vertex) of structural formula (molecular graph). In order to correlate ^{13}C NMR chemical shifts with the molecular structure we have to possess information about the environment of the given vertex. The chosen atom plays a role of the so-called root [146], a vertex distinguished from other vertices of the molecular graph. For alkanes embedding frequencies [147–149] specify the number of appearance of smaller rooted subtrees that are attached to the root of the given tree (alkane), see Figs. 4 and 5. Each atom (a non-equivalent vertex in the tree) in an alkane (tree) is determined by 13 descriptors $d=(d_1, d_2, \dots, d_{13})$ that are used as input activities of neural networks. The entry d_i determines the embedding frequency of the ith rooted subtree (Fig. 4) for the given rooted tree (the root is specified by that carbon atom of which the chemical shift is calculated). Their number and form are determined by our requirement to have all the rooted trees through 5 vertices. To avoid information redundancy, we have deleted those rooted trees, which embedding frequencies can be exactly determined from embedding frequencies of simpler rooted subtrees. This means, that we consider at most δ-carbon effects.

^{13}C NMR chemical shifts of all alkanes from C_2

[2] For details about this application see [145].

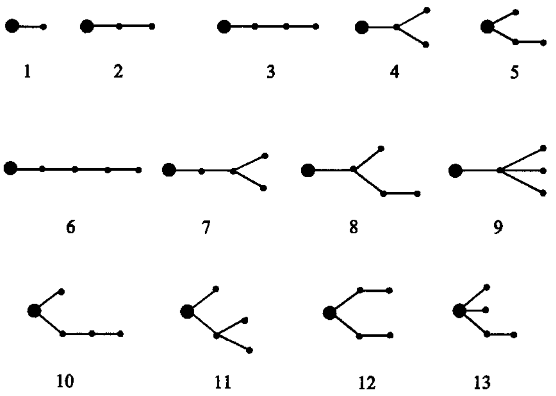

Fig. 4. List of 13 rooted subtrees that are used for the calculation of embedding frequencies.

to C_9 available in the book [150] (cf. Ref. [151]) (alkanes C_9 are not complete) are used as objects in our calculations. The total number of all alkanes considered in our calculations is 63, they give 326 different chemical shifts for topologically non-equivalent positions in alkanes. This set of 326 chemical shifts is divided into the training set and the test set.

The decomposition of whole set of chemical shifts into training and test sets was carried out by making use of the Kohonen neural network [4] with architecture specified by 14 input neurons and 15 × 15 = 275 output neurons situated on a rectangular grid 15 × 15.

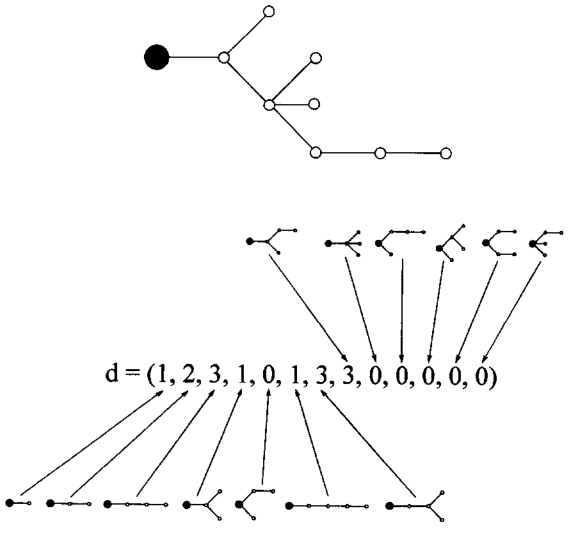

Fig. 5. Illustrative example of embedding frequencies of a rooted tree.

The input activities of each object (chemical shift) are composed of 14 entries, whereby the first 13 entries are embedding frequencies and the last, 14th entry, is equal to the chemical shift. Details of the used Kohonen network are described in Dayhoff's textbook [152]. We used Kohonen network with parameters $\alpha = 0.2$ (learning constant), $d_0 = 10$ (initial size of neighbourhood), and $T = 20\,000$ (number of learning steps). We have used the rectangular type of neighbourhood and the output activities were determined as L_1 (city-block) distances between input activities and the corresponding weights. After finishing the adaptation process, all 326 objects were clustered so that each object activates only one output neuron on the rectangular grid, and some output neurons are never activated and/or some output neurons are activated by one or more objects. This means that this decomposition of objects through the grid of output neurons may be considered as a clustering of objects, each cluster, composed of one or more objects, being specified by a single output neuron. Finally, the training set is created so that we shift one object (with the lowest serial index) from each cluster to the training set and the remaining ones to the test set. Then we get training set composed of 112 objects and the test set composed of 214 objects.

The results of our neural-network calculations for different numbers of hidden neurons (from one to five) are summarised in Table 1. The quantities SEC and R, are determined as follows

$$\mathrm{SEC}^2 = \frac{\Sigma_p (x_{\mathrm{obs}} - x_{\mathrm{calc}})^2}{N} \tag{17}$$

$$R^2 = 1 - \frac{\Sigma_p (x_{\mathrm{obs}} - x_{\mathrm{calc}})^2}{\Sigma_p (x_{\mathrm{obs}} - x_{\mathrm{mean}})^2} \tag{18}$$

We see that the best results are produced by the

Table 1
Results of neural-network calculations

Type of neural net.	Training set		Test set	
	SEC	R^2	SEC	R^2
(13,1,1)	1.1387	0.9976	1.1913	0.9837
(13,2,1)	0.9906	0.9980	1.0980	0.9957
(13,3,1)	0.8941	0.9998	1.0732	0.9966
(13,4,1)	0.7517	0.9999	1.0905	0.9946
(13,5,1)	0.6656	1.0000	1.1041	0.9944

Table 2
Results of LRA calculations

Type of LRA	Training set		Test set	
	SEC	R^2	SEC	R^2
All objects [a]	0.9994	0.9900	–	–
Training set	0.9307	0.9893	1.1624	0.9872

[a] Training set is composed of all 326 objects.

neural network (13,3,1) composed of three hidden neurons, its SEC value for objects from the test set being the lowest one. We can observe the following interesting property of feed-forward neural networks: The SEC value for training set monotonously decreases when the number of hidden neuron increases; on the other hand, the SEC value for test set has a minimum for three hidden neurons. This means that the predictability of neural networks for test objects is best for three hidden neurons, further increasing of their number does not provide better results for test set (this is the so-called overtraining).

In the framework of linear regression analysis (LRA) chemical shifts (in ppm units) are determined as a linear combination of all 13 descriptors plus a constant term

$$\delta = c_0 + \sum_{i=1}^{13} c_i d_i \tag{19}$$

Two different LRA calculations have been carried out. While the first calculation was based on the whole set of 326 objects (chemical shifts), the second calculation included only the objects from the training set (the same as for neural-network calculations). The obtained results are summarised in Table 2.

Comparing results of neural-network and LRA calculations, we see that the best neural-network calculation provides slightly better results for training objects than LRA. The SEC testing value for neural-network calculation is slightly smaller than it is for LRA calculation. Table 3 lists precision of predictions of chemical shifts. It means, for instance, that the neural-network (13,3,1) calculation for objects from the test set (eighth column in Table 3) provides the following prediction: for 74% (78% and 88%) of the shifts, the difference between the experimental and predicted values was less than 1.0 ppm (1.5 ppm and 2.0 ppm, respectively). On the other hand, what is very surprising, the LRA based on the training set gave slightly better prediction for test objects than the neural-network (13,3,1) calculation. Precision of predictions for differences 1.5 ppm and 2.0 ppm were slightly greater for LRA than for NN (neural network), see the sixth and eighth columns in Table 3.

As it is apparent from the results, the use of neural networks in this case is discutable, because it brings only the minimal advantages in comparing with linear regression analysis. This means that possible nonlinearities in the relationship between embedding frequencies and chemical shifts are of small importance. An effectiveness of neural-network calculations results from the fact that nonlinearities of input–output relationships are automatically taken into account. Since, as was mentioned above, nonlinearities in relationships between embedding frequencies and ^{13}C NMR chemical shifts in alkanes are of small (or negligible) importance, neural-network calculations could not provide considerably better results than LRA calculations. Finally, as a byproduct of our LRA calculations, we have obtained simple linear relationships between ^{13}C NMR chemical shifts

Table 3
Precision of prediction [a]

Prediction precision	Grant Ref. [15]	Lindeman Ref. [11]	LRA [b] all objects	LRA [c]		NN (13,3,1)	
				training	test	training	test
1.0 ppm	61%	61%	78%	78%	69%	87%	74%
1.5 ppm	77%	78%	89%	90%	85%	96%	78%
2.0 ppm	84%	89%	94%	97%	91%	98%	88%

[a] Rows indicate percentages of objects predicted by the given model with precision specified by maximum ppm absolute error shown in the first column.
[b] LRA which used all 326 objects for training set.
[c] LRA which used only 112 objects for training set.

in alkanes and embedding frequencies which are more precise (see Table 3) than similar relationships constructed by Grant [153] or Lindeman [151] often used in literature (cf. Ref. [150]).

10. Conclusions

ANNs should not be used without analysis of the problem, because there are many alternatives to the use of neural networks for complex approximation problems. There are obvious cases when the use of neural networks is quite inappropriate, e.g. when the system is described with the set of equations, that reflects its physico-chemical behaviour. ANNs is a powerful tool, but the classical methods (e.g. MLRA, PCA, cluster analysis, pattern recognition etc.) can sometimes provide better results in shorter time.

References

[1] W.S. McCulloch, W. Pitts, A logical calculus of ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.

[2] S. Haykin, Neural Networks – A Comprehensive Foundation, Macmillan, 1994.

[3] G.M. Maggiora, D.W. Elrod, R.G. Trenary, Computational neural networks as model-free mapping device, J. Chem. Inf. Comp. Sci. 32 (1992) 732–741.

[4] T. Kohonen, Self-organisation and Associative Memory, Springer Verlag, Berlin, 1988.

[5] J. Zupan, J. Gasteiger, Neural Networks for Chemists, VCH, New York, 1993.

[6] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.

[7] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, MA, 1996.

[8] A. Wieland, R. Leighton, Geometric analysis of neural network capabilities, in: 1st IEEE Int. Conf. on Neural Networks, Vol. 3, San Diego, CA, 1987, p. 385.

[9] W. Wu, B. Walczak, D.L. Massart, S. Heurding, F. Erni, I.R. Last, K.A. Prebble, Artificial neural networks in classification of NIR spectral data: Design of the training set, Chemom. Intell. Lab. Syst. 33 (1996) 35–46.

[10] M. Smith, Neural Networks for Statistical Modelling, Van Nostrand Reinhold, New York, 1993.

[11] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 (1992) 1–58.

[12] A. Blum, Neural Networks in C^{++}, Wiley, 1992.

[13] K. Hornik, Approximation capabilities of multi-layer neural networks, Neural Networks 4 (2) (1991) 251–257.

[14] K. Hornik, Some new results on neural network approximation, Neural Networks 6 (1993) 1069–1072.

[15] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford Univ. Press, Oxford, 1995.

[16] V. Kurkova, Kolmogorov's theorem and multilayer neural networks, Neural Networks 5 (3) (1992) 501–506.

[17] T. Masters, Practical Neural Network Recipes in C + +, Academic Press, 1993, p. 87.

[18] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge Univ. Press, Cambridge, 1996.

[19] S.M. Weiss, C.A. Kulikowski, Computer Systems That Learn, Morgan Kaufmann, 1991.

[20] M. Stone, Cross validation choice and assessment of statistical predictions, J. Roy. Statistical Soc. B36 (1974) 111–133.

[21] J.S.U. Hjorth, Computer Intensive Statistical Methods, Chapman and Hall, London, 1994.

[22] B. Efron, R.J. Tibshirani, An Introduction to the Bootstrap, Chapman and Hall, London, 1993.

[23] S. Amari, N. Murata, K.-R. Muller, M. Finke, H. Yang, Asymptotic statistical theory of overtraining and cross-validation, METR 95-06, Department of Mathematical Engineering and Information Physics, University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan, 1995.

[24] R. Andrews, J. Diedrich, A.B. Tickle, A survey and critiques for extracting rules from trained artificial neural networks, Internal printing of the Neurocomputing Research centre, Queensland University of Technology, Brisbane, 1995.

[25] J.W. Shavlik, A Framework for Combining Symbolic and Neural Learning, CS-TR-92-1123, November 1992, The University of Wisconsin. Available at http://www.cs.wisc.edu/trs.html.

[26] M.A. Franzini, Speech recognition with back propagation, Proc. IEEE 9th Annual Conf. Engineering in Medicine and Biology Society, Boston, MA, vol. 9, 1987, pp. 1702–1703.

[27] K. Matsuoka, J. Yi, Back-propagation based on the logarithmic error function and elimination of local minima, Proc. Int. Joint Conf. on Neural Networks, Singapore, vol. 2, 1991, pp. 1117–1122.

[28] S.A. Solla, E. Levin, M. Fleisher, Accelerated learning in layered neural networks, Complex Systems 2 (1988) 39–44.

[29] H. White, Learning in artificial neural networks: a statistical perspective, Neural Computation 1 (1989) 425–464.

[30] J.S. Bridle, Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters, in: D.S.Touretzky (Ed.), Advances in Neural Information Processing Systems, vol. 2, Morgan Kaufmann, San Maeto, CA, 1990, pp. 211-217.

[31] S.A. Solla, M.J. Holt, S. Semnani, Convergence of back-propagation in neural networks using a log-likelihood cost function, Electron. Lett. 26 (1990) 1964–1965.

[32] K. Matsuoka, A. van Ooyen, B. Nienhuis, Improving the convergence of the back-propagation algorithm, Neural Networks 5 (1992) 465–471.

[33] A.S. Weigend, D.E. Rumelhart, B.A. Hubermann, Back propagation, weight elimination and time series prediction, in: D.S. Touretzky, J.L. Elman, T.J. Sejnowski, G.E. Hinton (Eds.), Connectionist Models, Proc. 1990 Connectionist Models Summer School, Morgan Kaufmann, San Mateo, CA, 1991, pp. 105-116.

[34] J. Lee, Z. Bien, Improvement on function approximation capability of back-propagation neural networks, Proc. Int. Joint Conf. on Neural Network, Singapore, vol. 2, 1991, pp. 1367-1372.

[35] P.A. Shoemaker, M.J. Carlin, R.L. Shimabukuro, Back-propagation learning with trinary quantization of weight updates, Neural Networks 4 (1991) 231–241.

[36] T. Samad, Back-propagation improvements based heuristic arguments, Proc. Int. Joint Conf. on Neural Networks, Washington DC, vol. 1, 1990, pp. 565–568.

[37] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L. Jackel, Back-propagation applied to handwritten zip code recognition, Neural Computation 1 (1989) 541–551.

[38] J. Sietsma, R.J.F. Dow, Creating artificial neural networks that generalize, Neural Networks 2 (1991) 67–69.

[39] G. Teasuro, B. Janssens, Scaling relationships in back-propagation learning, Complex Systems 2 (1988) 39–44.

[40] J. Higashino, B.L. de Greef, E.H. Persoon, Numerical analysis and adaptation method for learning rate of back propagation, Proc. Int. Joint Conf. on Neural Networks, Washington DC, vol. 1, 1990, pp. 627–630.

[41] S.E. Fahlman, Fast-learning variations on back propagation: an empirical study, in: D.S. Touretzky, G.E. Hinton, T.J. Sejnowski (Eds.), Proc. 1988 Connectionist Models Summer School, Morgan Kaufmann, San Mateo, CA, 1989, pp. 38-51.

[42] R. Battiti, T. Tecchiolli, Learning with fast, second and no derivatives: a case study in high energy physics, Neurocomputing 6 (1994) 181–206.

[43] S.S. Rao, Optimisation: Theory and Applications, Ravi Acharya for Wiley Eastern, New Delhi, 1978.

[44] P.E. Gill, W. Murray, M. Wright, Practical Optimisation, Academic Press, London, 1981.

[45] C. deGroot, D. Wurtz, Plain back-propagation and advanced optimisation algorithms: a comparative study, Neurocomputing 6 (1994) 153–161.

[46] P.J.M. van Laarhoven, E.H.L. Aarts, Simulated Annealing. Theory and Applications, Reidel, Dordrecht, 1987.

[47] R.H.J.M. Otten, L.P.P.P. van Ginneken, Annealing Algorithm, Kluwer, Boston, 1989.

[48] V. Kvasnièka, J. Pospíchal, Augmented simulated annealing adaptation of feed-forward neural networks, Neural Network World 3 (1994) 67–80.

[49] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, D.L. Alkon, Accelerating the convergence of the back-propagation method, Biological Cybernetics 59 (1988) 257–263.

[50] D.V. Schreibman, E.M. Norris, Speeding up back-propagation by gradient correlation, Proc. Int. Joint Conf. on Neural Networks, Washington DC, vol. 1, 1990, pp. 723–736.

[51] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural networks 1 (1988) 226–238.

[52] T. Tollenaere, SuperSAB: fast adaptive back-propagation with good scaling properties, Neural Networks 3 (1990) 561–573.

[53] M. Riedmiller, H. Braun, A direct adaptive method for faster back-propagation learning: The RPROP algorithm, Proc. IEEE Int. Conf. on Neural Networks, San Francisco, 1993.

[54] J.R. Chen, P. Mars, Stepsize variation methods for accelerating the back-propagation algorithm, Proc. Int. Joint Conf. on Neural Networks, Portland, Oregon, vol. 3, 1990, pp. 601–604.

[55] P.P. van der Smagt, Minimisation methods for training feed-forward neural networks, Neural Networks 7 (1994) 1–11.

[56] E. Barnard, J.E.W. Holm, A comparative study of optimisation techniques for back-propagation, Neurocomputing 6 (1994) 19–30.

[57] K. Levenberg, A method for the solution of certain problems in least squares , Quart. Appl. Math. 2 (1944) 164–168.

[58] D. Marquardt, An algorithm for least-squares estimation of nonlinear parameters , SIAM J. Appl. Math. 11 (1963) 431–441.

[59] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Maquardt algorithm, IEEE Trans. Neural Networks 5 (6) (1995) 989–993.

[60] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.t. Vetterling, Numerical Recipes: The art of scientific computing, Cambridge, Cambridge Univ. Press, 1987. Also available on-line at http://cfatab.harvard.edu/nr/.

[61] E. Polak, Computational methods in optimisation, Academic Press, New York, 1971.

[62] M.J.D. Powell, Restart procedures for the conjugate gradient methods, Math. Prog. 12 (1977) 241–254.

[63] R. Fletcher, C.M. Reeves, Function minimization by conjugate gradients, Comput. J. 7 (1964) 149–154.

[64] E. Polak, G. Ribiere, Note sur la convergence de methods de directions conjures, Revue Francaise Information Recherche Operationnelle 16 (1969) 35-43.

[65] E. Barnard, Optimisation for training neural nets, IEEE Trans. Neural Networks 3 (1992) 232–240.

[66] M.F. Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.

[67] T.A. Andrea, H. Kalyeh, Application of neural networks in quantitative structure–activity relationships of dihydrofolate reductase inhibitors, J. Med. Chem. 33 (1990) 2583–2590.

[68] D. Manallack, D.J. Livingstone, Artificial neural networks: application and chance effects for QSAR data analysis, Med. Chem. Res. 2 (1992) 181–190.

[69] D.J. Livingstone, D.W. Salt, Regression analysis for QSAR using neural networks, Bioorg. Med. Chem. Let. 2 (1992) 213–218.

[70] C. Borggaard, H.H. Thodberg, Optimal minimal neural interpretation of spectra, Anal. Chem. 64 (1992) 545–551.

[71] I. Tetko, A.I. Luik, G.I. Poda, Application of neural networks in structure–activity relationships of a small number of molecules, J. Med. Chem. 36 (1993) 811–814.

[72] I.V. Tetko, D.J. Livingstone, A.I. Luik, Neural network studies 1: comparison of overfitting and overtraining, J. Chem. Inf. Comp. Sci. 35 (1995) 826–833.

[73] A.J. Miller, Subset selection in regression, Monographs on Statistics and Applied Probability, vol. 40, Chapmann and Hall, London, 1990.

[74] H. Akaike, A new look at statistical model identification, IEEE Trans. Automatic Control 19 (1974) 716–722.

[75] H. Lohninger, Feature selection using growing neural networks: the recognition of quinoline derivatives from mass spectral data, in: D. Ziessow (Ed.), Software Development in Chemistry 7, Proc. 7th CIC Workshop, Gosen/Berlin, 1992, GDCh, Frankfurt, 1993, p. 25.

[76] D. Rogers, A.J. Hopfinger, Application of genetic function approximation to quantitative structure–activity relationships and quantitative structure–property relationships, J. Chem. Inf. Comput. Sci. 34 (1994) 854–866.

[77] H. Kubinyi, Variable selection in QSAR studies-1. An evolutionary algorithm, Quant. Struc. Act. Relat. 13 (1994) 285–294.

[78] B.T. Luke, Evolutionary programming applied to the development of quantitative structure–activity and quantitative structure–property relationships, J. Chem. Inf. Comput. Sci. 34 (1994) 1279–1287.

[79] J.H. Wikel, E.R. Dow, The use of neural networks for variable selection in QSAR, Bioorg. Med. Chem. Let. 3 (1993) 645–651.

[80] I.V. Tetko, A.E.P. Villa, D.J. Livingstone, Neural network studies 2: variable selection, J. Chem. Inf. Comp. Sci. 36 (1996) 794–803.

[81] J. Leonard, K.A. Kramer, Improvement of back-propagation algorithm for training neural networks, Comput. Chem. Eng. 14 (1990) 337–341.

[82] Ch. Klawun, Ch.L. Wilkins, A novel algorithm for local minimum escape in back-propagation neural networks: application to the interpretation of matrix isolation infrared spectra, J. Chem. Inf. Comput. Sci. 34 (1994) 984–993.

[83] H. Lohninger, Evaluation of neural networks based on radial basis function and their application to the prediction of boiling points from structural parameter, J. Chem. Inf. Comp. Sci. 33 (1993) 736–744.

[84] J. Tetteh, E. Metcalfe, S.L. Howells, Optimisation of radial basis and back-propagation neural networks for modelling auto-ignition temperature by quantitative structure–property relationship, Chemom. Int. Lab. Syst 32 (1996) 177–191.

[85] A.U. Radomski, P. Jan, H. van Halbeek, B. Meyer, Neural network-based recognition of oligosaccharide ^{1}H-NMR spectra, Nat. Struct. Biol. 1-4 (1994) 217–218.

[86] U. Hare, J. Brian, J.H. Prestegard, Application of neural networks to automated assignment of NMR spectra of proteins, J. Biomol. NMR 4 (1) (1994) 35–46.

[87] A.U.R. Zamora, J.L. Navarro, F.J. Hidalgo, Cross-peak classification in two-dimensional nuclear magnetic resonance, J. Am. Oil Chem. Soc. 71 (1994) 361–364.

[88] A.U. Corne, A. Simon, J. Fisher, A.P. Johnson, W.R. Newell, Cross-peak classification in two-dimensional nuclear magnetic resonance spectra using a two-layer neural network, Anal. Chim. Acta 278 (1993) 149–158.

[89] Ch. Ro, R.W. Linton, New directions in microprobe mass spectrometry: molecular, microanalysis using neural networks, Microbeam Anal. (Deerfield Beach, FL) 1 (1992) 75–87.

[90] R. Goodacre, A. Karim, A.M. Kaderbhai, D.B. Kell, Rapid and quantitative analysis of recombinant protein expression using pyrolysis mass spectrometry and artificial neural networks: application to mammalian cytochrome b5 in *Escherichia coli*, B. J. Biotechnol. 34 (1994) 185–193.

[91] R. Goodacre, M.J. Neal, D.B. Kell, Rapid and quantitative analysis of the pyrolysis mass spectra of complex binary and tertiary mixtures using multivariate calibration and artificial neural networks, Anal. Chem. 66 (1994) 1070–1085.

[92] J. Gasteiger, X. Li, V. Simon, M. Novic, J. Zupan, Neural nets for mass and vibrational spectra, J. Mol. Struct. 292 (1993) 141–159.

[93] W. Werther, H. Lohninger, F. Stancl, K. Varmuza, Classification of mass spectra. A comparison of yes/no classification methods for the recognition of simple structural properties, Chemom. Intell. Lab. Syst. 22 (1994) 63–76.

[94] T. Visser, H.J. Luinge, J.H. van der Maas, Recognition of visual characteristics of infrared spectra by artificial neural networks and partial least squares regression, Anal. Chim. Acta 296 (1994) 141–154.

[95] M.K. Alam, S.L. Stanton, G.A. Hebner, Near-infrared spectroscopy and neural networks for resin identification, Spectroscopy, Eugene, Oregon, 9 (1994) 30, 32–34, 36–38, 40.

[96] D.A. Powell, V. Turula, J.A. de Haseth, H. van Halbeek, B. Meyer, Sulfate detection in glycoprotein-derived oligosaccharides by artificial neural network analysis of Fourier-transform infrared spectra, Anal. Biochem. 220 (1994) 20–27.

[97] K. Tanabe, H. Uesaka, Neural network system for the identification of infrared spectra, Appl. Spectrosc. 46 (1992) 807–810.

[98] M. Meyer, K. Meyer, H. Hobert, Neural networks for interpretation of infrared spectra using extremely reduced spectral data, Anal. Chim. Acta 282 (1993) 407–415.

[99] J.M. Andrews, S.H. Lieberman, Neural network approach to qualitative identification of fuels and oils from laser induced fluorescence spectra, Anal. Chim. Acta 285 (1994) 237–246.

[100] B. Walczak, E. Bauer-Wolf, W. Wegscheider, A neuro-fuzzy system for X-ray spectra interpretation, Mikrochim. Acta 113 (1994) 153–169.

[101] Z. Boger, Z. Karpas, Application of neural networks for interpretation of ion mobility and X-ray fluorescence spectra, Anal. Chim. Acta 292 (1994) 243–251.

[102] A. Bos, M. Bos, W.E. van der Linden, Artificial neural networks as a multivariate calibration tool: modeling the iron–chromium–nickel system in X-ray fluorescence spectroscopy, Anal. Chim. Acta 277 (1993) 289–295.

[103] S. Iwasaki, H. Fukuda, M. Kitamura, High-speed analysis technique for gamma-ray and X-ray spectra using an asso-

ciative neural network, Int. J. PIXE, Volume Date 3 (1993) 267–273.

[104] S. Iwasaki, H. Fukuda, M. Kitamura, Application of linear associative neural network to thallium-activated sodium iodide gamma-ray spectrum analysis, KEK Proc. 1993, 93–98, 73–83.

[105] M.N. Souza, C. Gatts, M.A. Figueira, Application of the artificial neural network approach to the recognition of specific patterns in Auger electron spectroscopy, Surf. Interf. Anal. 20 (1993) 1047–1050.

[106] H.G. Schulze, M.W. Blades, A.V. Bree, B.B. Gorzalka, L.S. Greek, R.F.B. Turner, Characteristics of back-propagation neural networks employed in the identification of neurotransmitter Raman spectra, Appl. Spectrosc. 48 (1994) 50–57.

[107] M.J. Lerner, T. Lu, R. Gajewski, K.R. Kyle, M.S. Angel, Real time identification of VOCs in complex mixtures by holographic optical neural networking (HONN), Proc. Electrochem. Soc., 1993, pp. 93–97; Proc. Symp. on Chemical Sensors II, 1993, pp. 621–624.

[108] X. Ni, Y. Hsia, Artificial neural network in Mossbauer spectroscopy, Nucl. Sci. Tech. 5 (1994) 162–165.

[109] W.L. Morgan, J.T. Larsen, W.H. Goldstein, The use of artificial neural networks in plasma spectroscopy, J. Quant. Spectrosc. Radiat. Transfer 51 (1994) 247–253.

[110] N. Sreerama, R.W. Woody, Protein secondary structure from circular dichroism spectroscopy. Combining variable selection principle and cluster analysis with neural network, ridge regression and self-consistent methods, J. Mol. Biol. 242 (1994) 497–507.

[111] B. Dalmas, G.J. Hunter, W.H. Bannister, Prediction of protein secondary structure from circular dichroism spectra using artificial neural network techniques, Biochem. Mol. Biol. Int. 34 (1994) 17–26.

[112] S.L. Thaler, Neural net predicted Raman spectra of the graphite to diamond transition, Proc. Electrochem. Soc., 1993, pp. 93–117; Proc. 3rd Int. Symp. on Diamond Materials, 1993, pp. 773–778.

[113] D.L. Clouser, P.C. Jurs, Simulation of ^{13}C nuclear magnetic resonance spectra of tetrahydropyrans using regression analysis and neural networks, Anal. Chim. Acta 295 (1994) 221–231.

[114] A. Panaye, J.P. Doucet, B.T. Fan, E. Feuilleaubois, S.R.E. Azzouzi, Artificial neural network simulation of ^{13}C NMR shifts for methyl-substituted cyclohexanes, Chemom. Intell. Lab. Syst. 24 (1994) 129–135.

[115] Y. Miyashita, H. Yoshida, O. Yaegashi, T. Kimura, H. Nishiyama, S. Sasaki, Non-linear modelling of ^{13}C NMR chemical shift data using artificial neural networks and partial least squares method, THEOCHEM 117 (1994) 241–245.

[116] C. Affolter, J.T. Clerc, Prediction of infrared spectra from chemical structures of organic compounds using neural networks, Chemom. Intell. Lab. Syst. 21 (1993) 151–157.

[117] P. Bhagat, An introduction to neural nets, Chem. Eng. Prog. 86 (1990) 55.

[118] J.C. Hoskins, D.M. Himmelblau, Artificial neural network models of knowledge representation in chemical engineering, Comput. Chem. Eng. 12 (1988) 881.

[119] S.N. Kavuri, V. Venkatasubramanian, Using fuzzy clustering with ellipsoidal units in neural networks for robust fault classification, Comput. Chem. Eng. 17 (1993) 765–784.

[120] C. Puebla, Industrial process control of chemical reactions using spectroscopic data and neural networks: A computer simulation study, Chemom. Intell. Lab. Syst. 26 (1994) 27–35.

[121] K. Ye, K. Fujioka, K. Shimizu, Efficient control of fed-batch bakers' yeast cultivation based on neural network, Process Control Qual. 5 (1994) 245–250.

[122] I. Grabec, W. Sachse, D. Grabec, Intelligent processing of ultrasonic signals for process control applications, Mater. Eval. 51 (1993) 1174–1182.

[123] N. Qian, T.J. Sejnowski, Predicting the secondary structure of globular proteins using neural network models, J. Mol. Biol. 202 (1988) 568–584.

[124] M. Vieth, A. Kolinski, J. Skolnick, A. Sikorski, Prediction of protein secondary structure by neural networks: encoding short and long range patterns of amino acid packing, Acta Biochim. Pol. 39 (1992) 369–392.

[125] F. Sasagawa, K. Tajima, Toward prediction of multi-states secondary structures of protein by neural network, Genome Inf. Ser. (1993), 4 (GENOME informatics workshop IV), 197–204.

[126] J. Sun, L. Ling, R. Chen, Predicting the tertiary structure of homologous protein by neural network method, Gaojishu Tongxun 1 (1991) 1–4.

[127] D.T. Manallack, T. David, D.D. Ellis, D.J. Livingstone, Analysis of linear and non-linear QSAR data using neural networks, J. Med. Chem. 37 (1994) 3758–3767.

[128] R.D. King, J.D. Hirst, M.J.E. Sternberg, New approaches to QSAR: neural networks and machine learning, Perspect. Drug Discovery Des. 1 (1993) 279–290.

[129] D.J. Livingstone, D.W. Salt, Regression analysis for QSAR using neural networks, Bioorg. Med. Chem. Lett. 2 (1992) 213–218.

[130] S. Nakai, E. Li-Chan, Recent advances in structure and function of food proteins: QSAR approach, Crit. Rev. Food Sci. Nutr. 33 (1993) 477–499.

[131] W.G. Richards, Molecular similarity, Trends QSAR Mol. Modell. 92, Proc. Eur. Symp. Struct. Act. Relat.: QSAR Mol. Modell., C.G. Wermuth (Ed.), 9th ed., ESCOM, Leiden, 1992, pp. 203–206.

[132] V.S. Rose, A.P. Hill, R.M. Hyde, A. Hersey, pK_a prediction in multiply substituted phenols: a comparison of multiple linear regression and back-propagation, Trends QSAR Mol. Modell. 92, 1993; Proc. Eur. Symp. Struct. Act. Relat.: QSAR Mol. Modell., 9th (1993).

[133] M. Chastrette, J.Y. De Saint Laumer, J.F. Peyraud, Adapting the structure of a neural network to extract chemical information. Application to structure–odor relationships, SAR QSAR Environ. Res. 1 (1993) 221–231.

[134] D. Domine, J. Devillers, M. Chastrette, W. Karcher, Estimating pesticide field half-lives from a back-propagation neural network, QSAR Environ. Res. 1 (1993) 211–219.

[135] B. Cambon, J. Devillers, New trends in structure–biodegradability relationships, Quant. Struct. Act. Relat. 12 (1993) 49–56.

[136] G. Kateman, Neutral networks in analytical chemistry?, Chemom. Intell. Lab. Syst. 19 (1993) 135–142.

[137] J.R.M. Smits, W.J. Melssen, G.J. Daalmans, G. Kateman, Using molecular representations in combination with neural networks. A case study: prediction of the HPLC retention index, Comput. Chem. 18 (1994) 157–172.

[138] Y. Cai, L. Yao, Prediction of gas chromatographic retention values by artificial neural network, Fenxi Huaxue 21 (1993) 1250–1253.

[139] A. Bruchmann, P. Zinn, Ch.M. Haffer, Prediction of gas chromatographic retention index data by neural networks, Anal. Chim. Acta 283 (1993) 869–880.

[140] D.A. Palmer, E.K. Achter, D. Lieb, Analysis of fast gas chromatographic signals with artificial neural systems, Proc. SPIE Int. Soc. Opt. Eng. (1993) 1824; Applications of Signal and Image Processing in Explosives Detection Systems, 109–119.

[141] W.C. Rutledge, New sensor developments , ISA Trans. 31 (1992) 39–44.

[142] V. Sommer, P. Tobias, D. Kohl, Methane and butane concentrations in a mixture with air determined by microcalorimetric sensors and neural networks, Sens. Actuators B 12 (1993) 147–152.

[143] C. Di Natale, F.A.M. Davide, A. D'Amico, W. Goepel, U. Weimar, Sensor arrays calibration with enhanced neural networks, Sens. Actuators B 19 (1994) 654–657.

[144] B. Hivert, M. Hoummady, J.M. Henrioud, D. Hauden, Feasibility of surface acoustic wave (SAW) sensor array processing with formal neural networks, Sens. Actuators B 19 (1994) 645–648.

[145] D. Svozil, J. Pospíchal, V. Kvasnièka, Neural-network prediction of carbon-13 NMR chemical shifts of alkanes, J. Chem. Inf. Comput. Sci. 35 (1995) 924–928.

[146] F. Harary, Graph Theory, Addison- Wesley, Reading, MA, 1969.

[147] R.D. Poshusta, M.C. McHughes, Embedding frequencies of trees, J. Math. Chem. 3 (1989) 193–215.

[148] M.C. McHughes, R.D. Poshusta, Graph-theoretic cluster expansion. Thermochemical properties for alkanes, J. Math. Chem. 4 (1990) 227–249.

[149] V. Kvasnièka, J. Pospíchal, Simple construction of embedding frequencies of trees and rooted trees, J. Chem. Inf. Comp. Sci. 35 (1995) 121–128.

[150] H.O. Kalinowski, S. Berger, S. Braum, ^{13}C NMR Spektroskopie, G. Thieme, Stuttgart, 1984.

[151] L.P. Lindeman, J.Q. Adams, Carbon-13 nuclear magnetic resonance spectroscopy: chemical shifts for the paraffins through C_9, Anal.Chem. 43 (1993) 1245–1252.

[152] J. Dayhoff, Neural Network Architectures, Van Nostrand-Reinhold, New York, 1990.

[153] D.M. Grant, E.G. Paul, Carbon-13 magnetic resonance II. Chemical shift data for the alkanes, J. Am. Chem. Soc. 86 (1964) 2984–2990.