Evaluation of Neural Networks Based on Radial Basis Functions and Their Application to the Prediction of Boiling Points from Structural Parameters

H. Lohninger†

Institute for General Chemistry, Technical University Vienna, Lehargasse 4/152, A-1060 Vienna, Austria

Received April 26, 1993*

The performance of neural networks based on radial basis functions (RBF neural networks) is evaluated. The network uses modified Gaussian kernel functions which have shown better results for classification purposes. RBF networks are tested for the variation of some design parameters and the presence of noise in the sample data. The problems of generalization and extrapolation are addressed, and a procedure is suggested for how to test for the generalization ability of neural networks. An RBF network has been applied to chemical data in order to estimate boiling points at normal pressure from structural parameters. The results show a significant decrease of the prediction error when compared to results obtained by multiple linear regression.

INTRODUCTION

Neural networks have become a universal tool for establishing a nonlinear mapping of an n-dimensional space $\mathcal{R}^n$ to a p-dimensional space $\mathcal{R}^p$. There exist many models of neural networks which have different approaches both in architecture and in learning algorithms. Presently the most widely used network type is a multilayer perceptron which is trained by the back-propagation learning algorithm. Although this algorithm has already been published in the mid-70s,[1] a breakthrough of the application of neural networks has taken place only after this training algorithm had been re-invented 10 years later.[2] One of the major drawbacks of the back-propagation algorithm comes from the fact that (1) it can be caught in local minima during learning and (2) it exhibits very long training times which makes it unsuitable for all applications where a large number of training runs have to be performed (e.g. feature selection[3]).

More recently neural networks gain more importance in chemistry too. Neural networks are applied to problems of spectroscopy, QSAR, secondary structure of proteins, and multivariate calibration, which are the main areas of interest up to now. A comprehensive review on neural networks and their application in chemistry is given by Zupan and Gasteiger.[4]

This paper will show a network model which has the advantage of small training times[5] and is guaranteed to reach the global minimum of the error surface during training. Furthermore, this model allows an easier understanding and analysis of the results since the underlying idea is simple and the training algorithm is based on well-understood calculus. The model is based on radial basis functions and is therefore also called RBF network. The present work evaluates the performance of RBF networks and introduces a small modification which enhances its performance in classification tasks. In addition, an attempt will be made to describe the inner mechanics of radial basis function networks by some kind of visualization. An RBF network is applied to a simple problem of structure–property relationship, and its results are compared to those published by other authors using multiple linear regression.

THEORY

Since a thorough mathematical description of RBF networks is given elsewhere,[6-8] the theoretical background will be presented only to an extent which is necessary to explain the results obtained in this work. RBF networks have a special architecture in that they have only three layers (input, hidden, output) and there is only one layer where the neurons exhibit a nonlinear response. Furthermore, other authors have suggested the inclusion of some extra neurons which serve to calculate the reliability of the output signals.[9] Figure 1 shows the structure of an RBF network as it is used in this paper.

The input layer has—as in any other network model—no calculating power and serves only to distribute the input data among the hidden neurons. The hidden neurons exhibit a more complicated transfer function than for example in back-propagation networks. The output neurons in turn have a linear transfer function which makes it possible to calculate simply the optimum weights associated with these neurons. An extra neuron is used in this work to detect whether extrapolation occurs.

As Specht[10] pointed out, RBF networks fall between regression models and nearest neighbor classification schemes, which can be looked upon as content addressable memories. Furthermore the behavior of an RBF network can be controlled by a single parameter which determines if the network behaves more like a multiple linear regression or more like a content addressable memory. The regression methods usually use all available data to build a model. If additional data are presented to this model, the whole model has to be recalculated. On the other hand memories have a special storage location for each data sample presented, so the model does not have to be changed at all for additional data; only additional memory cells have to be provided. RBF networks fall between since additional data effect only those functions which are localized closely to the incoming data. This property is of special importance in networks that adjust their response during operation ("drawing attention", see ref 7).

RBF neural networks belong to the class of kernel estimation methods. These methods use a weighted sum of a finite set of nonlinear functions $\Phi_i(\mathbf{x}-\mathbf{c})$ to approximate an unknown function $f(\mathbf{x})$. The approximation is constructed from the data samples presented to the network using eq 1, where h is

$$f(\mathbf{x}) = \sum_{i=1}^{h} w_i \Phi(\mathbf{x}-\mathbf{c}_i) \qquad (1)$$

the number of kernel functions, $\Phi(\cdot)$ is the kernel function, $\mathbf{x}$ is the input vector, $\mathbf{c}$ is a vector which represents the center

† Electronic mail address: hlohning@email.tuwien.ac.at.

* Abstract published in *Advance ACS Abstracts,* August 15, 1993.

0095-2338/93/1633-0736$04.00/0

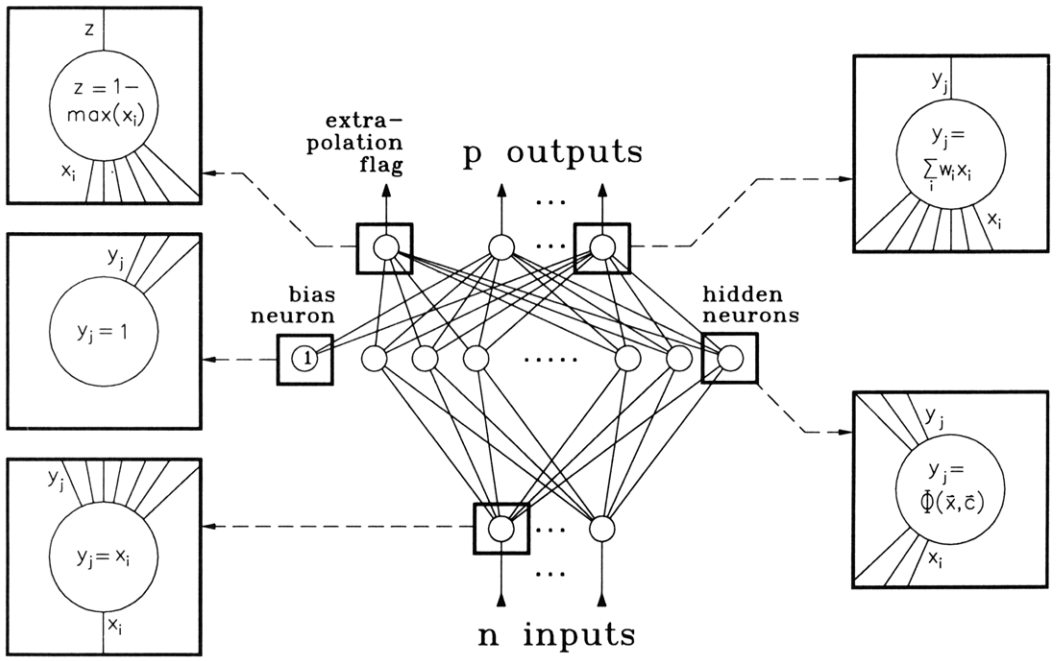

Figure 1. Structure of the RBF network used in this paper. The network consists of three layers and an extra neuron to flag the state of extrapolation.

of the kernel function in the n-dimensional space, and w_i are the coefficients to adapt the approximating function $f(\mathbf{x})$. If these kernel functions are mapped to a neural network architecture, a three layered network can be constructed where each hidden node is represented by a single kernel function (see Figure 1) and the coefficients w_i of eq 1 represent the weights of the output layer.

The type of each kernel function can be chosen out of a large class of functions,[11] and in fact is has been shown more recently that an arbitrary nonlinearity is sufficient to represent any functional relationship by a neural network.[12-15] Gaussian kernel functions are widely used throughout the literature. As will be shown in this work, a small modification to the Gaussian kernel function improves the performance of RBF networks for classification tasks. Therefore a modified Gaussian kernel function according to eq 2 was used in the present work, where

$$\Phi(\cdot) = \frac{1 + R}{R + \exp[S(\mathbf{x}-\mathbf{c})^{\mathrm{T}}\mathbf{A}(\mathbf{x}-\mathbf{c})]} \qquad (2)$$

S is a system parameter which determines the "slope" of each kernel function and thus the smoothness of the estimation and R is a parameter which "flattens" the function around the center (Figure 2). For $R = 0$, eq 2 reduces to the commonly used Gaussian kernel. The vector $\mathbf{x}$ holds the input data, and the vector $\mathbf{c}$ represents the position of the center of the kernel function. The matrix $\mathbf{A}$ serves to normalize the metrics of the input data space. If $\mathbf{A}$ is an identity matrix, then the Euclidean distances between the centers $\mathbf{c}$ and the input vectors are used. If the matrix $\mathbf{A}$ contains the reciprocal standard deviations on its diagonal with all other elements equal to zero, the data are scaled to have unit variance. It is interesting to note that if matrix $\mathbf{A}$ equals the inverse of the covariance matrix of the local data (i.e. data which are closely located to the center of a specific kernel function), the Mahalanobis distance comes into effect for the distance between $\mathbf{c}$ and $\mathbf{x}$.

An RBF network consists of a system of h (h is the number of hidden neurons) nonlinear equations which is fortunately linear in its parameters w_i (eq 1). In order to solve this system one has to find appropriate values for the parameters $\mathbf{c}_i$, R, S, $\mathbf{A}$, and w_i.

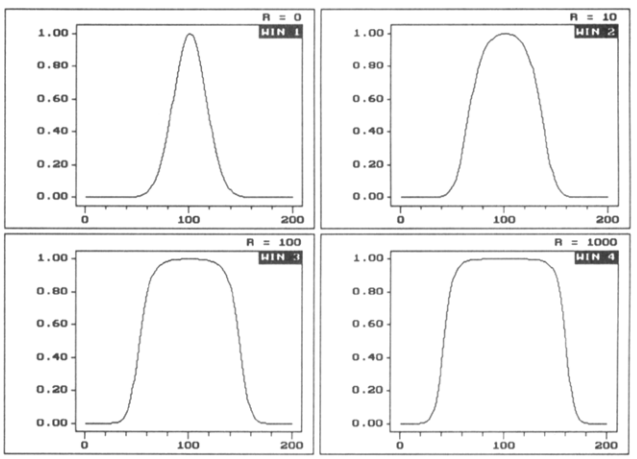

Figure 2. One-dimensional kernel function with $S = 0.002$ and $R = 0$, 10, 100, 1000, respectively.

The number of hidden neurons h strongly influences the generalization properties of a network. It can be shown that the calculated response of an RBF network will perfectly match all data points in the training set if h equals the number of training objects. This is clearly unfavorable since no generalization can take place under these conditions. The number of hidden neurons should therefore be kept as low as possible, and the trained network must be tested against a test set in order to determine its performance. If the data set is too small, the addition of noise can be used to check the reliability of the approximated function (see below).

If we assume that the four parameters $\mathbf{c}$, R, S, and $\mathbf{A}$ are fixed or can be determined by some method, the problem reduces to a system of linear equations, which is usually overdetermined:

$$\mathbf{y} = \mathbf{D}\mathbf{w} \qquad (3)$$

where $\mathbf{y}$ is the vector of the target values of all samples, $\mathbf{w}$ is the vector of the weights w_i, and $\mathbf{D}$ is a design matrix whose elements are defined according to eq 2 and which is augmented by a column vector with all elements set to 1. This additional vector provides the signal of the bias neuron (cf. Figure 1).

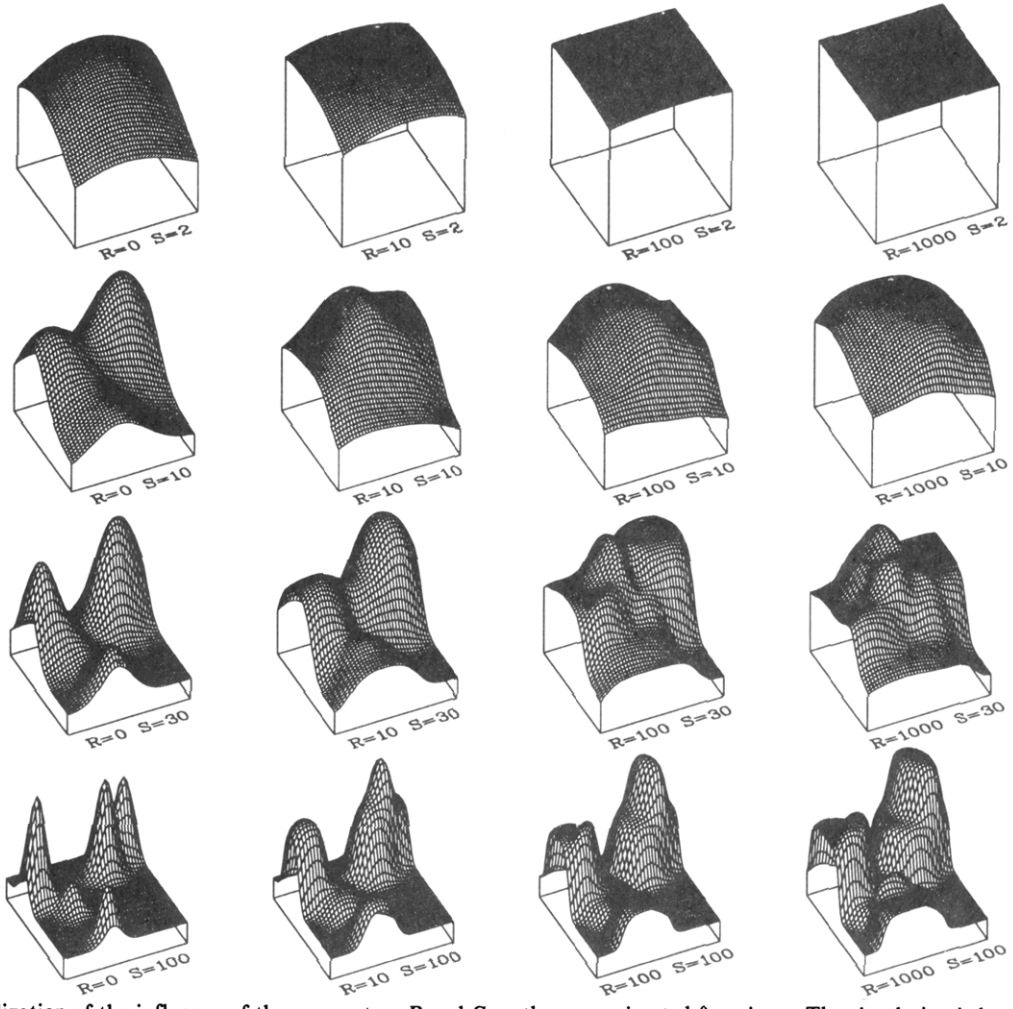

Figure 3. Visualization of the influence of the parameters R and S on the approximated functions. The simulation is based on five hidden neurons. The positions of the five corresponding kernel functions can be seen best in the lower left corner of the figure.

These equations can be solved for $\mathbf{w}$ in a straightforward manner, by calculating the pseudo-inverse using singular value decomposition:[16]

$$\mathbf{w} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{y} \tag{4}$$

However there exist other techniques of solving eq 4 which become important for systems with a large number of hidden neurons. One of these techniques is the application of Hebb's rule during the learning process, which however does not yield an exact solution.

The only problem which remains is to find some procedure in order to determine $\mathbf{c}_i$, R, S, and $\mathbf{A}$ of eq 2. This is not a trivial problem, and several procedures are proposed in the literature.[17-20] The best way is certainly to adjust these parameters by data-driven processes, but there is no well-established way to do this and one has to make a compromise between ease (and time) of processing and the quality of the results.

These parameters can be found in a practical manner by a two-step process. First we have to fix the positions $\mathbf{c}_i$ of the kernel functions. This can be done in several ways, for example, by a cluster analysis, a Kohonen network,[21] or a genetic algorithm.[22] The idea behind this step is to put together several neighboring points in the input data space and to represent these points by a single RBF. In the second step the parameters R and S and the matrix $\mathbf{A}$ of the particular kernel function should be adjusted according to the shape and width of the data space represented by the neighboring points. The algorithm used in this work is described below in the Discussion section.

Geometrical Interpretation of a Radial Basis Function Estimation. The following short example will give a summary of the features of radial basis function networks in a geometrical form. Suppose one has to approximate an arbitrary three-dimensional surface (think of a part of the Alps in Austria), which is specified by some sample points. Now the task of the network training is to accomplish a setup of the centers $\mathbf{c}_i$, the weights w_i, the matrix $\mathbf{A}$, and the parameters S and R such that the deviation of the estimated function from the sample points is minimum.

In the three-dimensional case the kernel function can be visualized by mountains (or hills) whose shapes are controlled by the parameters R and S and the matrix $\mathbf{A}$. S controls the steepness of the hill, R controls its top flatness, and $\mathbf{A}$ controls the shape of the basis of that hill (i.e. the eccentricity of the resulting ellipse). The weights w_i determine the height of the mountains, and the centers $\mathbf{c}_i$ control the positions of these peaks. The sum of all the kernel functions gives the approximation of the three-dimensional surface. Figure 3 shows the resulting surface for five hidden nodes with arbitrary but fixed parameters $\mathbf{c}_i$, w_i, and $\mathbf{A}$ and with varying parameters S and R. As one can see, the variations of R and S create a wide range of possible shapes from single peaks ($R = 0$, $S = 100$) to an almost planar surface ($R = 1000$, $S = 2$).

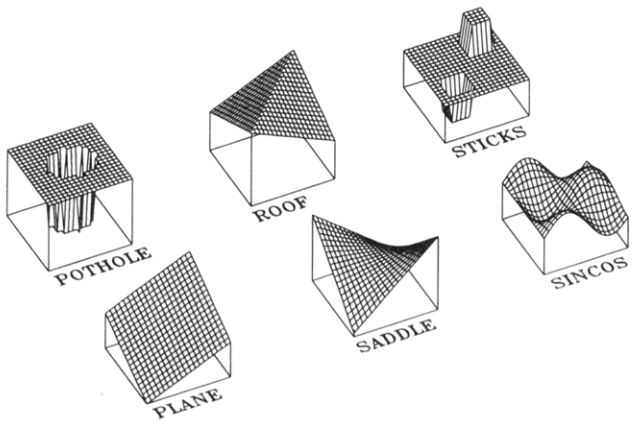

Figure 4. 3D plot of the test data.

DISCUSSION

Data Sets. In order to evaluate the function of RBF networks, six artificial data sets were set up. These data sets were chosen rather arbitrarily but reflect several different types of functions, which may be important in practical applications. All of the data sets were created for a two-dimensional input space and a one-dimensional output space and cover a square of 20 × 20 units. The input data points are positioned on integer grid points ranging from 0 to 19 on each side. Thus it is easy to visualize the results (Figure 4). Experiments have shown that the results presented here by using two-dimensional data can also be applied to data spaces of higher (or lower) dimensionality.

Data Set 1 (*PLANE*). As neural networks use nonlinear transfer functions, it is not trivial that neural networks can approximate planes as well as nonlinear surfaces. In order to account for this issue, a simple linear relation is used to establish a planar surface:

$$y = (x_1 + x_2)/2 \tag{5}$$

Data Set 2 (*SADDLE*). A problem which is commonly used in neural computation to test the performance of neural networks is the exclusive-or problem. The data set SADDLE reflects this type of problems for continuous function mapping since each pair of opposite corners of the data set produces similar output values and each pair of adjacent corner produces different outputs:

$$y = 0.1(x_1 - 10)(x_2 - 10) \tag{6}$$

Data Set 3 (*SINCOS*). These data represent a complex continuous surface and can be calculated according to eq 7.

$$y = 5 \sin(0.1\pi x_1) \cos(0.1\pi x_2) \tag{7}$$

Data Set 4 (*POTHOLE*). These data show a typical situation which is met when data have to be classified. The output variable has a constant value a_1 for all data except in an elliptic region where the output variable has a constant value a_2. The data are defined according to eq 8.

$$y = +8, \quad \text{for } ((x_1 - 7)^2 + (x_2 - 10)^2)^{1/2} + ((x_1 - 13)^2 + (x_2 - 10)^2)^{1/2} \leq 10$$
$$-8, \quad \text{for all other input data} \tag{8}$$

Data Set 5 (*ROOF*). As neural networks create continuous function mappings a strong test case should be a data set where the first derivative does not exist on some points. The data set ROOF consists of two intersecting planes which create a shape like a roof of a house.

$$y = 10 - \text{abs}(9.5 - x_1/2 - x_2/2) \tag{9}$$

Data Set 6 (*STICKS*). This is another data set which addresses the domain of classification. Two comparatively small rectangular regions of the data space have outputs which lie above and below the bulk of the other data.

$$y = +8, \quad \text{for } 12 \leq x_i \leq 16$$
$$-8, \quad \text{for } 4 \leq x_i \leq 7$$
$$0, \quad \text{for all other data samples} \tag{10}$$

Determining the Positions of the Centers of Radial Basis Functions. As already mentioned above, the positioning of the radial basis functions is of great importance for the performance of the neural network. In this paper the centers of the RBFs are determined by a simple clustering algorithm. This algorithm repeatedly replaces those two data points which have the smallest distance by their center of gravity until the number of the remaining data points equals the predefined number of hidden nodes. These remaining data points establish the centers of the RBFs of the hidden nodes.

There is one drawback when applying any cluster algorithm to the input data in order to determine the RBF centers. The calculated positions reflect only clusters of the input data space which may not be appropriate in classification problems. Some authors circumvent this drawback by using clustering algorithms which account for the class of the data points.[19] But these algorithms are designed for classification purposes and do not work for continuous modeling. Especially in cases where the training data contain some contradictions, the resulting RBF centers are not optimal.

In order to circumvent this difficulty another method is suggested in this work: The cluster analysis is applied to a composite data space which is spanned by both the input and the target variables and (if available) by the class number. This automatically solves the above mentioned problems and works equally well for classification tasks. Therefore all available variables are scaled to have zero mean and unit variance. Then the composite data space is used for cluster analysis. After the clustering has been completed, the resulting positions are mapped into the unscaled input data space in order to determine the centers of the kernel functions.

Determining the Best Values for Parameters *S* and *R*. In order to investigate the dependence of the quality of estimation on the parameters S and R, RBF networks were trained with systematically varied parameters R and S. R was varied from 0.0 to 20.0 with a step width of 0.5, and S was varied from 0.5 to 20.0 with the same step width. This gave a grid of 1640 points which established a rectangular map of 41 × 40 base points. The square of the correlation coefficient between target and calculated value was used to characterize the quality of the fit. Figures 5 and 6 show these maps for two representative cases, one for continuous function estimation (SINCOS) and one for classification tasks (STICKS).

The interpretation of this experiment leads to two conclusions which have to be kept in mind when configuring RBF networks:

(1) The optimum parameters S and R are problem dependent and there is no fixed rule for selecting optimum values, although some authors give hints to possible ways for the selection of the best parameters.[23] Up to now it is certainly best to scan R and S systematically since RBF networks can be trained rather fast, especially if one is aware that the positions of the kernel functions need to be calculated only once for this type of investigation. A possibly better approach

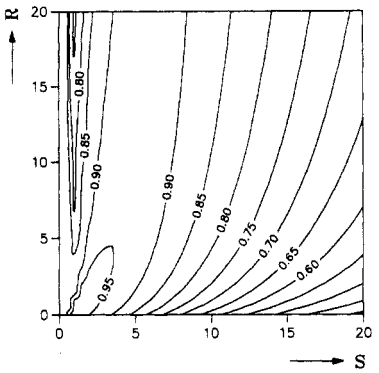

Figure 5. Contour lines of the square of the correlation coefficient for the data set SINCOS. Best results are obtained for low values of R and S.

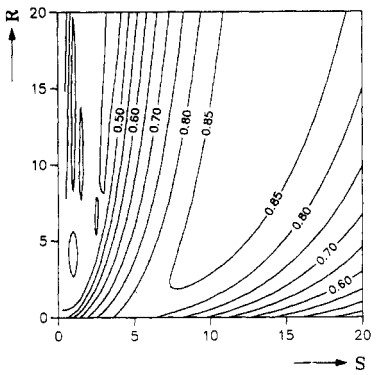

Figure 6. Contour lines of the square of the correlation coefficient for the data set STICKS. Best results are obtained for high values of R and medium values of S.

would be to determine the values of R and S individually for each kernel function, but no feasible procedure is known so far to the author of this paper.

(2) Classification tasks can be solved better by using higher values for R and S whereas continuous function approximation performs usually best with low values of R and S (R near to or equal to 0.0, S around 0.5). This can be easily explained in a figurative way. As classification tasks require a response which is constant within a class and changes very rapidly at class boundaries, it is intuitively clear that using kernel functions which have a flat region at the center and a steep region at their rims provides a better estimation of such functions.

Dependence of the Performance on the Number of Hidden Units. The number of hidden neurons h greatly influences the performance of a neural network. If the number of hidden nodes is too low, the network cannot calculate a proper estimation of the data. On the other hand, if too many hidden neurons are used, the network tends to overfit the training data. In order to show the influence of the number of hidden units, two experiments were set up—one that can be looked upon as the worst case of function approximation and one which is based on the data set defined above.

For the first experiment, a data set of 80 samples with three variables was used. These data consisted of evenly distributed random numbers in all three variables. A network was trained to approximate one of the three variables as a function of the other two. The design parameters S and h were changed systematically in a range of S = 0.1–5.0 and h = 1–80. The results are shown in Figure 7. As can be seen, the goodness of the fit increases almost linearly with the number of hidden neurons from 0.0 (no match) to 1.0 (perfect match) if the parameter S is high enough. The reason for the fact that a

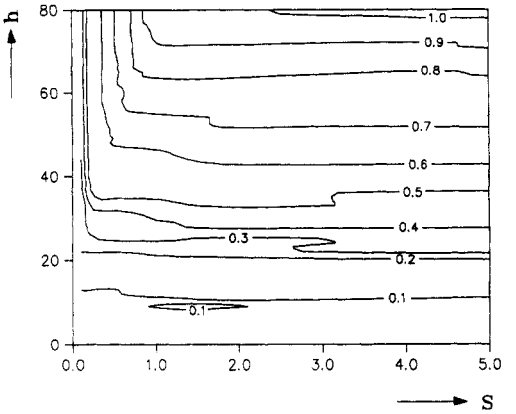

Figure 7. Goodness of fit for a varying number of hidden neurons and varying parameter S when applied to a set of 80 random data. The parameters R was held constant at 0.0.

perfect match can be obtained only if S is high enough lies in the characteristics of the kernel functions. If S is low, a single kernel function is activated by a wider area of the input data space and is therefore influenced by more than one data sample. Since the data in a random set are uncorrelated, there is only a small chance that these samples will not be contradictory. So the network can match the training data only if a single data point activates one and only one kernel function. Consequently, if this condition is fulfilled, the network matches only a fraction of the sample data. This fraction is proportional to the ratio of the number of hidden neurons and the number of training data.

The second experiment uses the data sets specified above and an additional random data set, where the data to be estimated are evenly distributed random numbers in the range of 0.0–1.0. Each data set has been trained with a varying number of hidden neurons using the optimum values for the parameters S and R in each training. The results are shown in Figure 8. As can be easily seen, there are some striking differences when compared to the results of the first experiment.

First, the networks reach their optimum performance by using only a few (3–18) hidden neurons. This early increase in the estimation goodness (when compared to the random data set) is due to the correlation of neighboring data points. A completely uncorrelated data set, like the random data set, will give a perfect match only when the number of hidden neurons equals the number of data points (400 in this example).

Second, the data sets which test for the classification performance (POTHOLE and STICKS) reach only a near-optimum estimation performance with a few hidden neurons (5 and 9, respectively). Thereafter the estimation performance increases very slowly until it becomes 1.0 (perfect match). This effect heavily depends on the distance of the classes in the input variable space. If the border region between the classes is about the same size as the distance of the data points within each class, the network cannot learn a perfect match with only a few hidden neurons. If the border region is large enough (larger than the range of sensitivity of a single kernel function), a perfect match can be calculated using only a few neurons (theoretically this could be achieved by using one hidden neuron less than the number of classes, if each class cluster lies within a hypersphere and the spheres do not overlap).

Problem of Generalization. Since neural networks are very powerful in estimating virtually any function given by some data points, a major problem arises from this fact. If the parameters of a network are chosen in a wrong way, a situation

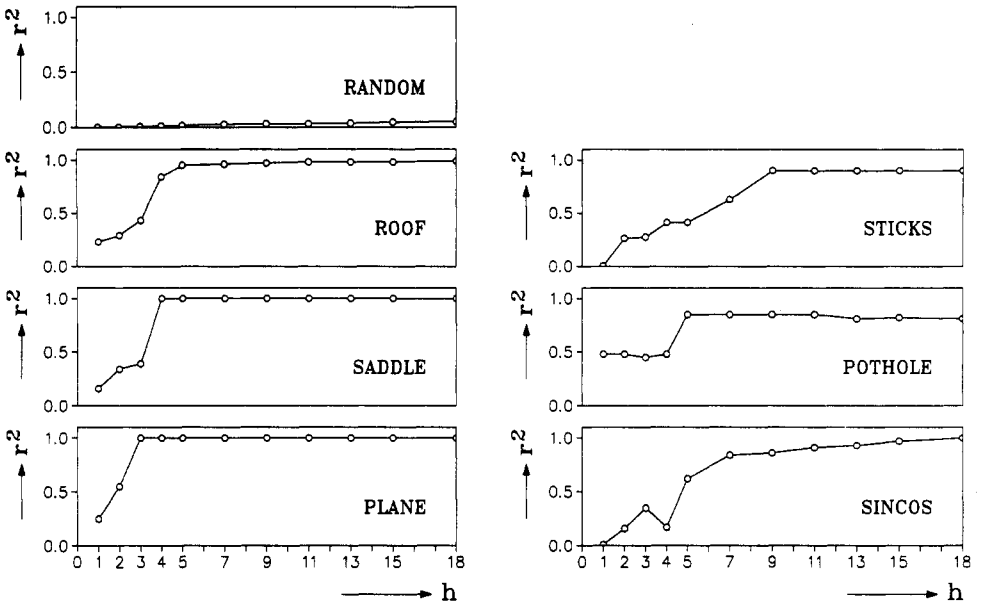

Figure 8. Goodness of fit for a varying number of hidden neurons.

will arise where the resulting approximation is too good. This means that the approximation reflects the noise rather than the underlying trend in the data. This effect is also sometimes referred to as overtraining. Of course this situation is clearly unfavorable and has to be avoided. As Maggiora et al.[24] stated, there is no known procedure for measuring the degree of generalization for continuous function mappings.

One possible procedure for estimating the extent of generalization is to train the network by using several copies of the data set with different amounts of noise added. Therefore two measures are defined: the goodness of fit of the estimation (square of correlation coefficient between sample and estimated data), $r^2_{t,e}$; the square of the correlation coefficient between the estimated data of the original data set and the estimated data calculated from the noisy data, $r^2_{e0,en}$.

These figures are calculated for various levels of noise. The trends of these two figures as noise goes up indicate the generalization of the network. A network which performs well will show a decreasing $r^2_{t,e}$ since the increasing level of noise will not be reflected in the estimated function. On the other hand the value of $r^2_{e0,en}$ should stay almost constant since the estimated function of a noisy data set will not differ much from the estimated function of the original data set. The situation is just a mirror image for networks where overfitting occurs: the parameter $r^2_{t,e}$ will be almost constant and the value of $r^2_{e0,en}$ will decrease with increasing noise since the networks tend to adjust themselves to the noisy sample data neglecting the underlying trend of the data.

In order to verify the suggested procedure, the following test setup was chosen: The two data sets SINCOS and STICKS served as the basis for these investigations, the first as a model of continuous function approximation, the second as a model of a classification problem. From each of these two basic data sets, three experiments were constructed which differed only in the number of data points used (400, 200, and 100) and the number of hidden neurons (15, 38, and 70, respectively). The data were reduced by regularly omitting grid points, thus maintaining the shape of the test surfaces. The first case (400 data, 15 hidden neurons) should result in a good generalization, the second case (200 data, 38 hidden neurons) should be somewhat worse, and the third case (100 data, 70 hidden neurons) should give almost go generalization as the number of hidden neurons comes close to the number of data points. The parameters R and S were optimally chosen for each experiment by scanning a wide range of possible values. The data were superimposed with an increasing amount of mean-centered white noise with the noise amplitude A_n normalized to the variance of the target data. For each level of noise an RBF network was trained to give an estimation of the underlying function, and the values of $r^2_{t,e}$ and $r^2_{e0,en}$ were calculated. The results are shown in Figure 9. As can be seen from these results a network with a few hidden neurons (curves A in Figure 9) gives a good estimation of the underlying data up to a level of noise which equals the signal variance (A_n = 1.0). As the number of hidden neurons goes up and the number of available data points decreases, the networks tend to lose generalization power. In these cases (curves C) even a small amount of noise shows a significant decrease in estimation quality.

Extrapolation. Neural networks exhibit a major drawback when compared to linear methods of function approximation: they cannot extrapolate. This is due to the fact that a neural network can map virtually any function by adjusting its parameters according to the presented training data. For regions of the variable space where no training data are available, the output of a neural network is not reliable.

In order to overcome this problem one should in some form register the range of the variable space where training data are available. In principle this could be done by calculating the convex hull of the training data set. If unknown data presented to the net are within this hull, the output of the net can be looked upon as reliable. However, the concept of the convex hull is not satisfactory since this hull is complicated to calculate and provides no solution for problems where the input data space is concave. A better way proposed by Leonard et al.[9] is to estimate the local density of training data by using Parzen windows.[25] This would be suitable for all types of networks. Radial basis function networks provide another elegant yet simple method of detecting extrapolation regions.

Since the centers of the kernel functions are positioned to represent several input vectors the activation of the kernel functions (eq 2) can be used to detect whether unknown data lie within the range of the training set. In order to get a single parameter which flags an extrapolation condition, the difference of the maximum of the activations of all kernel functions to 1.0 should be used (cf. Figure 1). The output

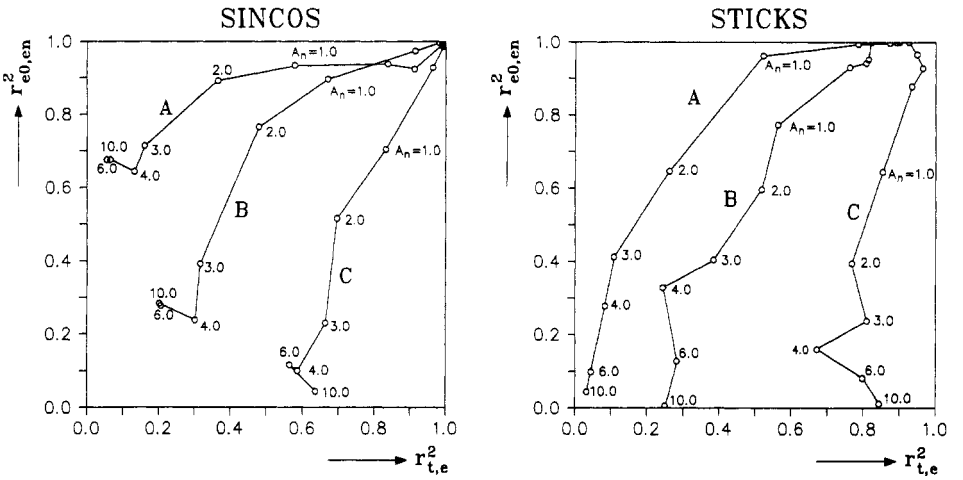

Figure 9. Dependence of $r^2_{t,e}$ and $r^2_{e0,en}$ on various levels of added noise A_n is shown for three networks of different size and generalization capability. The results have been obtained using the data sets SINCOS (left) and STICK (right) as basis: curve A, 400 data points, 15 hidden neurons; curve B, 200 data points, 38 hidden neurons; C, 100 data points, 70 hidden neurons.

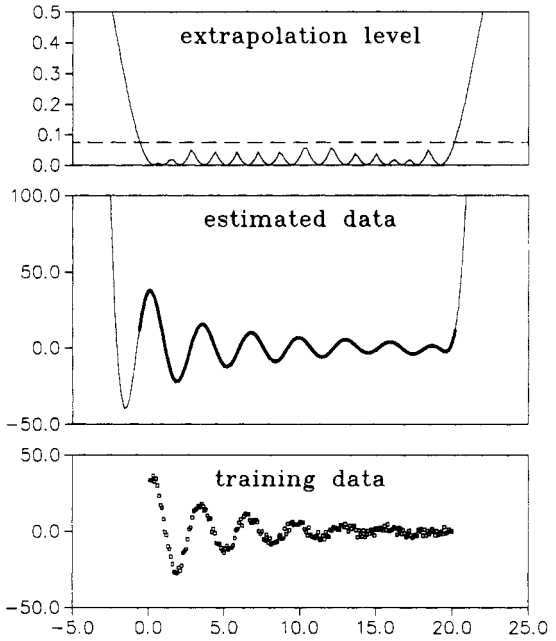

Figure 10. Estimation of a damped oscillation and the level of extrapolation. The bold part of the estimated data indicates the range where data are considered to be reliable.

values of a neural network should be considered to be unreliable or wrong when this extrapolation parameter increases above a certain limit (usually around 0.1).

In order to show this, a simple one-dimensional experiment is set up: 200 data points which are sampled at equidistant intervals from a noisy signal originating from a damped oscillation (Figure 10, lower part) are used as training data. An RBF network consisting of 15 hidden neurons is trained to estimate the underlying function of these data. Then the trained network is tested with input values which scan a larger area of values than that used during the training. The response of the network is shown in the middle part of Figure 10. One can easily see that the output values of the network are only valid within the range of the training data. Outside this range the response rapidly drifts off. In the upper part of Figure 10 the output of the extrapolation neuron is shown. If a threshold value is used as indicated by the dashed line, the state of extrapolation can be detected easily. The corresponding part of the estimated function which can be looked upon as reliable is shown as a bold line in the middle part.

Table 1. Comparison of Multiple Linear Regression and Neural Networks[a]

		experiment 1		experiment 2	
		linear regression	RBF network	linear regression	RBF network
training	r^2	0.9714 [26]	0.9853	0.9737	0.9900
	s_{err}	8.2 [26]	5.8	7.8	4.9
cross-validation	r^2	0.9702	0.9795	0.9728	0.9845
	s_{err}	8.3	6.9	8.0	5.9

[a] The table shows the goodness of the fit and the standard deviation of the estimation error for both training and cross-validation runs.

APPLICATION TO CHEMICAL DATA

In order to show the application of neural networks to chemical data, recently published results by Balaban et al.[26] on the correlation of normal boiling points and structural parameters of 185 ethers, peroxides, acetals, and their sulfur analogs were used as a basis for further investigations using neural networks. Balaban et al. used multiple linear regression (MLR) to set up a correlation between boiling points and three structural parameters:

$^1\chi$	the Randic index[27]
N_S	number of sulfur atoms in the compound
J_{het}	the modified connectivity index[26]

They came up with a regression equation which produced a goodness of fit of 0.9714 and a standard deviation of the error of the estimated boiling points of 8.2 °C.

In order to evaluate the performance of radial basis function networks, two experiments were performed: First, the data set of Balaban was used to approximate the correlation of normal boiling points to chemical structures using an RBF network. Secondly, simple structural parameters which are partly the same as those used by Balaban were calculated and supplied to the RBF network. The trained networks were subjected to cross-validation using the leave-a-quarter-out approach. The results obtained by the neural networks are summarized in Table I.

In the first experiment 20 hidden neurons have been used. The best values for the parameters R (0.0) and S (0.02) have been found by scanning a larger range of R and S systematically. The matrix **A** was set up as a diagonal matrix whose diagonal elements were equal to the reciprocals of the standard deviations of the input variables. The results show an increase

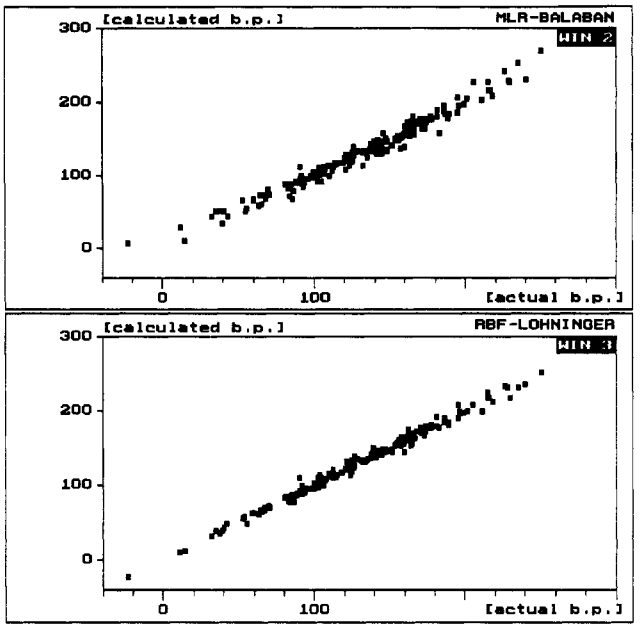

Figure 11. Comparison of Results of multiple linear regression (top) and RBF neural networks (bottom). The two charts show the estimated boiling points versus the actual boiling points.

in the predicted precision of the boiling points when compared to the multiple linear regression (s_{err} = 6.9 vs 8.3 °C).

The second experiment was carried out by setting up a network with 20 hidden neurons and using the parameters R and S found in the first experiment. The input variables for the network were selected from a pool of 11 simple topological and structural parameters. The following parameters were available:

N_C	number of carbon atoms
N_O	number of oxygen atoms
N_S	number of sulfur atoms
N_{het}	number of heteroatoms in molecule
TDia	topological diameter
TRad	topological radius
$^1\chi$	Randic Index[27]
$^1\chi_{mod}$	modified Randic Index (see below)
J	topological parameter defined by Balaban[28]
J_{het}	parameter defined by Balaban[26]
N_{met}	number of methyl groups

The parameter $^1\chi_{mod}$ is a modified Randic index and is defined by

$$^1\chi_{mod} = {}^1/_2 \sum_{i=1}^{N} \sum_{j=1}^{b_i} a_i/(b_i c_j)^{1/2} \qquad (11)$$

where N is the number of non-hydrogen atoms in the molecule, a_i is the atomic number of atom i, b_i is the number of bonds (vertex degree in the hydrogen-depleted graph) of atom i, and c_j is the number of bonds of the neighboring atoms of atom i.

Out of these eleven parameters three variables were selected by a technique previously called "growing neural network"[3] to give the best estimation of the boiling points: N_O, $^1\chi$, and $^1\chi_{mod}$. The training of a neural network with 20 hidden neurons and parameters R = 0.0 and S = 0.02 resulted in a standard deviation of the estimation error which was about 1 °C lower than that of the first experiment. The results of both Balaban's and this author's work are shown in Figure 11.

As mentioned above, the results of neural networks have to be cross-validated. In order to become comparable, the regression results have also been cross-validated, although the ratio of variables to the number of data is sufficiently low. From the comparison of the results presented in Table I one can draw two conclusions. First the cross-validated results of the neural network were about 1 °C worse than the estimated data, whereas the regression results exhibit almost no increase in prediction error. This is a direct consequence of the relation of modeling power and generalization ability.

The second aspect of the results in Table I concerns the approximation of nonlinearities. If the results of experiment 1 and experiment 2 are compared, it can be seen that selecting other input variables does not yield a significant decrease of the prediction error in the case of linear regression but does so with the RBF network. The inspection of the correlation of actual and estimated MLR data of experiment 2 shows that these variables yield a slightly nonlinear relation, which of course is better matched by the neural network.

CONCLUSION

Neural networks based on radial basis functions are a valuable tool in function estimation and can be looked upon as universal approximations. They exhibit a high speed of learning when compared to multilayer perceptron trained by the back-propagation algorithm. They can be applied equally well to classification problems, especially if the kernel functions are adjusted for that purpose. RBF networks excel in the point that they provide means for detecting the level of extrapolation. However further work has to be done on the problem of generalization and on the positioning and shaping of the kernel functions.

ACKNOWLEDGMENT

The author wishes to thank the German "Bundesministerium für Forschung und Technologie" for supporting this work by granting a research stay at the Technical University of Munich, FRG.

REFERENCES AND NOTES

(1) Werbos, P. Beyond regression: new tools for prediction and analysis in behavorial sciences. Ph.D. Thesis, Harvard University, Cambridge, MA, Aug 1974.
(2) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Nature* **1986**, *332*, 533–536.
(3) Lohninger, H. Feature Selection Using Growing Neural Networks: The Recognition of Quinoline Derivatives from Mass Spectral Data. In *Proceedings of the 7th CIC-Workshop*, Gosen/Berlin, Nov 1992; Ziessov, D., Ed.; Springer: Berlin, in press.
(4) Zupan, J.; Gasteiger, J. Neural networks: A new method for solving chemical problems or just a passing phase? *Anal. Chim. Acta* **1991**, *248*, 1–30.
(5) Specht, D. F.; Shapiro, P. D. Training speed comparison of probabilistic neural networks with back propagation networks. In *Proceedings of the International Neural Network Conference*, Paris, France; Kluwer: Dordrecht, The Netherlands, 1990; Vol. 1, pp 440–443.
(6) Broomhead, D. S.; Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex Syst.* **1988**, *2*, 312–355.
(7) Jokinen, P. A. Dynamically Capacity Allocating Neural Networks for Continuous Learning Using Sequential Processing of Data. *Chemom. Intell. Lab. Syst.* **1991**, *12*, 121–145.
(8) Moody, J.; Darken, C. Fast learning in networks of locally-tuned processing units. *Neural Comput.* **1989**, *1*, 281–294.
(9) Leonard, J. A.; Kramer, M. A.; Ungar, L. H. A neural network architecture that computes its own reliability. *Comput. Chem. Eng.* **1992**, *16* (9), 819–835.
(10) Specht, D. F. Probabilistic neural networks. *Neural Networks* **1990**, *3*, 109–118.
(11) Micchelli, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constr. Approximations* **1986**, *2*, 11–22.
(12) Kreinovich, V. Y. Arbitrary nonlinearity is sufficient to represent all functions by neural networks: A theorem. *Neural Networks* **1991**, *4*, 381–383.
(13) Funahashi, K. On the approximate realization of continuous mappings by neural networks. *Neural Networks* **1989**, *2*, 183–192.

(14) Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Networks* **1989**, *2*, 359–366.
(15) Hartman, E.; Keeler, J. D.; Kowalski, J. M. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Comput.* **1990**, *2*, 210–215.
(16) Golub, G. H.; Kahan, W. Calculating the singular values and pseudo-inverse of a matrix. *J. SIAM Numer. Anal. Ser. B* **1965**, *2*, 205–224.
(17) Chen, S.; Cowan, C. F. N.; Grant, P. M. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks* **1991**, *NN-2*, 302–309.
(18) Musavi, M. T.; Chan, K. H.; Hummels, D. M.; Kalantri, K.; Ahmed, W. A probabilistic model for evaluation of neural network classifiers. *Pattern Recognit.* **1992**, *25*, 1241–1251.
(19) Musavi, M. T.; Ahmed, W.; Chan, K. H.; Faris, K. B.; Hummels, D. M. On the training of radial basis function classifiers. *Neural Networks* **1992**, *5*, 595–603.
(20) Weymaere, N.; Martens, J.-P. A fast and robust learning algorithm for feedforward neural networks. *Neural Networks* **1991**, *4*, 361–369.
(21) Kohonen, T. *Self-Organization* and *Associative Memory*; Springer: Berlin, 1989.
(22) Goldberg, D. E. *Genetic algorithms in search, optimization, and machine learning*; Addison-Wesley: New York, 1989.
(23) Bishop, C. Improving the generalization properties properties of radial basis function neural networks. *Neural Comput.* **1991**, *3*, 579–588.
(24) Maggiora, G. M.; Elrod, D. W.; Trenary, R. G. Computational neural networks as model-free mapping devices. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 732–741.
(25) Parzen, E. On estimation of a probability density function and mode. *Ann. Math. Stat.* **1962**, *33*, 1065–1076.
(26) Balaban, A. T.; Kier, L. B.; Joshi, N. Correlations between chemical structure and normal boiling points of acyclic ethers, peroxides, acetals and their sulfur analogues. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 237–244.
(27) Randic, M. On characterization of molecular branching. *J. Am. Chem. Soc.* **1975**, *97*, 6609.
(28) Balaban, A. T. Highly discriminating distance-based topological index. *Chem. Phys. Lett.* **1992**, *89*, 399–404.