Chemometrics and Intelligent Laboratory Systems 28 (1995) 49–60

Chemometrics and intelligent laboratory systems

Robustness analysis of radial base function and multi-layered feed-forward neural network models

E.P.P.A. Derks [a,*], M.S. Sánchez Pastor [b], L.M.C. Buydens [a]

[a] *Laboratory for Analytical Chemistry, Faculty of Science, Catholic University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands*

[b] *Department of Mathematical Analysis, Faculty of Science and Food Technology and Chemistry, University of Burgos, Apdo. 231, 09080 Burgos, Spain*

Received 11 August 1994; accepted 21 November 1994

Abstract

In this paper, two popular types of neural network models (radial base function (RBF) and multi-layered feed-forward (MLF) networks) trained by the generalized delta rule, are tested on their robustness to random errors in input space. A method is proposed to estimate the sensitivity of network outputs to the amplitude of random errors in the input space, sampled from known normal distributions. An additional parameter can be extracted to give a general indication about the bias on the network predictions. The modelling performances of MLF and RBF neural networks have been tested on a variety of simulated function approximation problems. Since the results of the proposed validation method strongly depend on the configuration of the networks and the data used, little can be said about robustness as an intrinsic quality of the neural network model. However, given a data set where 'pure' errors from input and output space are specified, the method can be applied to select a neural network model which optimally approximates the nonlinear relations between objects in input and output space. The proposed method has been applied to a nonlinear modelling problem from industrial chemical practice. Since MLF and RBF networks are based on different concepts from biological neural processes, a brief theoretical introduction is given.

1. Introduction

In the last five years there has been a tremendous 'hype' about neural networks. Most publications appeared in fields of multivariate calibration, classification, temporal pattern recognition, signal processing, control image processing, data compression and knowledge processing [1,2]. Neural networks are mainly used as nonlinear function approximators and classifiers. They are nonparametric and generally little knowledge has to be incorporated in the modelling process.

Although neural network theory is based on the concept of biological neural processes, criticism has arisen claiming that traditional methods for statistics and pattern recognition can replace neural networks or should at least be as effective. Next to that neural networks are often criticized for their lack of predictability. Little theory has been developed for the estimation of confidence intervals on network predictions, which often makes the method unacceptable for industrial applications.

* Corresponding author.

SSDI 0169-7439(94)00082-4

The main reason for this criticism is probably based on the fact that neural networks are frequently applied on problems which could also be solved by traditional statistical methods. Since these traditional techniques are easy to apply and theoretically better founded, they should be preferred.

However, neural networks still offer solutions which cannot be as well obtained by conventional methods. Especially when adaptive algorithms are required and a lot of training or calibration samples are available (e.g. process control), neural networks can offer good solutions.

Since the introduction of neural networks in chemistry, a considerable number of papers has emerged in various chemical magazines, showing powerful modelling performances in situations where no analytical model could be derived. Neural networks are able to model any relation to the desired degree of accuracy without any knowledge about the internal relations within the data.

Special attention has to be payed to the bias–variance trade-off. The root mean squared error (RMSE) consists of a bias and variance term which work in opposite directions. Overtraining the neural networks will cause a decrease in bias (accuracy) but there is no real gain since the precision of future network predictions will be very poor. This phenomenon is in 'the neural network society' also known as the memory effect. An optimal compromise between bias and variance can be obtained by computational expensive crossvalidation procedures.

Still, after applying crossvalidation or leave-one-out methods (LOOM), little can be said about the predictability of the neural models in terms of confidence intervals. In most chemical applications, signals are disturbed by pink noise or interference noise from environmental sources [3]. Hence, the obtained neural network model trained with the noisy data patterns will also contain some uncertainty, proportional to the observational noise. So the variance of output errors depends on the observational noise emanating from the apparatus or environment and the 'noise' in the model.

Prior empirical studies about the effect of the numbers of layers in a neural network indicated that the predictive ability decreases proportionally with the number of layers, since errors are accumulated through the network layers. It has originally been proven by the mathematician Kolmogorov [4] that any continuous function can be implemented exactly by a three-layered feed-forward neural network having k input units, $(2k+1)$ hidden units in the middle layer, and m units in the top layer. Hence, the enhancement of the capacity of a neural network should be carried out in one single layer by increasing the number of hidden units. Consequently, in this paper, three-layered network structures have been used.

As previously mentioned in this text, little theory has been derived for the estimation of confidence intervals on network predictions. If no analytical solution can be derived, the predictive ability can be estimated by Monte Carlo (MC) simulations. During these simulations, the deviation of the total error in the computed output as a result of deviations in input space is investigated. Input errors are sampled from known distributions and propagated to the output of the network. The weights of the network can also contain some uncertainty. Consequently, these weight errors need also to be known in order to estimate the prediction errors of the neural model. However, statistically, little can be said about weight errors since the generalized delta learning rule does not yield reproducible weights as a result of multiple local minima in the error hyperplane and different training conditions. In this work, the assumption has been made that the weight errors do not contribute to the neural model. This firm assumption does not allow us to use the validation method for quantitative purposes. However, if the weight errors (model noise) are small, good qualitative judgements about the neural network model can be obtained.

In Section 3, a method based on Monte Carlo simulations is presented, for estimating error probability density functions (PDFs) of output units of radial base function (RBF) and multi-layered feed-forward (MLF) neural network models. Trained neural networks (with 'infinite' precision) have been used. The MC simulations have been repeated for a number of decreasing noise amplitudes, covering the range of interest. The initial input errors are chosen by sampling from predefined and, speaking chemically, representative input error distributions. The input distributions are generated using prior information about pure errors obtained from replicated measurements. The MC simulations are used in order to quantify the sensitivity of output units to the amplitude of random input errors. Note that only the amplitude of the random input errors is addressed (denoted by the symbol Ω) and that all input errors are controlled by this one factor.

The relevance of these experiments becomes apparent when inverse simulations are carried out. This means that PDFs of input variables are estimated for a given PDF of the output units. Inverse simulations can be used in order to optimize experiments with respect to precision criteria. Hence, variables can be identified which are responsible for poor predictions.

2. Theory

2.1. Feed-forward networks

Both MLF and RBF networks are known to be able to approximate any continuous function to some degree of accuracy [5–8].

The choice of a specific type of neural network for function approximation problems depends on a variety of factors and is usually related to the available prior information, desired accuracy and tractability of the network. In this paper, the robustness to the input errors for both types of networks is investigated as an additional quality factor for a neural network model.

For both networks, the network topology (i.e. number of layers) has been configured identically. This way, at least the number of network parameters and the size of the network remains comparable. The basic differences between the two methods are to be found in the activation or base functions. The feed-forward structure of both types of neural networks is depicted in Fig. 1, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)^{\mathrm{T}}$ and $\boldsymbol{h} = (h_1, h_2, \ldots, h_l)^{\mathrm{T}}$ represent an input vector and the hidden units respectively. The output of the network can be computed by a linear combination of base functions or hidden units, specified by $\boldsymbol{h}$. The base functions or hidden units are discussed in the following two sections.

Both RBF and MLF can be trained by means of gradient descent methods [9,10]. For the MLF network, gradient descent methods like the generalized delta learning rule have become mandatory, based on the fact that the MLF parameters appear in a nonlinear fashion. However, good results have been obtained by training based on optimization methods like

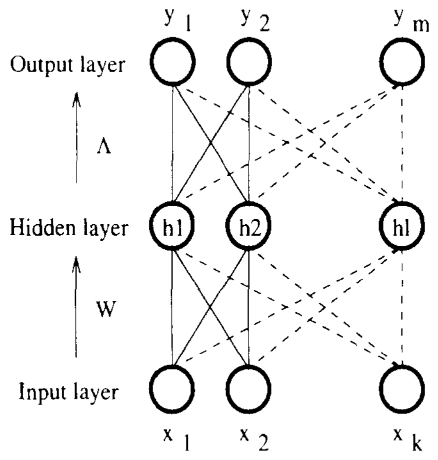

Fig. 1. The feed-forward structure of MLF and RBF neural network models. The weight matrix ($\boldsymbol{W}$) is used to calculate the input of the neurons in the hidden layer denoted by $\boldsymbol{h}$, by means of the product $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}_l$ and the Euclidean distance $\|\boldsymbol{x} - \boldsymbol{W}_l\|$ respectively. In the latter case, the matrix $\boldsymbol{W}$ contains the centers for the radial base functions. The network output is obtained by a linear combination of the output of the hidden units or base functions weighted by the matrix $\boldsymbol{\Lambda}$.

simulated annealing [11,12] and genetic algorithms [13]. Also modifications of the generalized delta learning rule and training methods based on extended Kalman filters [14] have shown good convergence properties.

In contrast to MLF, the parameters of the RBF network can be adjusted by linear learning methods like Kohonen or Hebbian learning. Linear parameter adaption yields faster learning and higher probability to converge to the global minimum. The Kohonen learning rule has the additional advantage that the parameters of the RBF network can be adjusted locally. This means that only the parameters of the best-matching base function are adapted without distorting the other parameters. These properties make it possible to learn new objects without forgetting the old ones. Since no distortion of neighbouring parameters is allowed, the training time will be shortened significantly.

In order to establish a uniform notation for the whole text, let $\boldsymbol{x} = (x_{ik})$ represent the input pattern matrix formed by vectors in the d-dimensional input space, $y_i \in \mathbb{R}$ the corresponding desired output (target output) and $\hat{y}_i$ the response computed for the network. This way, the function f to be approximated can be considered as a set of N data points in input space $\mathbb{R}^d$ such that $f(\boldsymbol{x}_i)=y_i$ for $i=1, 2, \ldots, N$. During training, the network tries to approximate y_i as close as possible.

2.2. The MLF network

The backpropagation training rule for MLF neural networks as described in the classical paper by Rumelhart et al. [2], has attracted a great deal of interest in recent years and has been successfully used in a large variety of applications [10,15]. As far as the network topology is concerned, it is possible to have several hidden layers, connections that skip over layers and lateral or even recurrent connections. Although these advanced topics are important, they tend to obfuscate the simplicity of the algorithm. Hence, the primary description of the MLF network assumes a three-layered feed-forward topology with only one hidden layer containing sigmoidal shaped activation functions, as is shown in Fig. 1. The input and output neurons only act as flow-through units. The equation for each output unit of the MLF network is given by

$$y_i=\sum_{j=1}^{L} \boldsymbol{\lambda}_j\, \psi\left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{W}_j\right) \tag{1}$$

where $\boldsymbol{W}_j$ represents the connection weights between the input and hidden layer, $\boldsymbol{x}_i^{\mathrm{T}}$ the transpose of input vector $\boldsymbol{x}_i$ and $\boldsymbol{\lambda}_j$ the coefficients assigned to the linear combination of the hidden units.

The activation function $\psi(x)$ can be any continuous function for which a derivative can be obtained, such as the hyperbolic tangent or the sigmoid function which is defined by

$$\psi\left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{W}_j\right)=\frac{1}{1+\exp\left(-\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{W}_j\right)} \tag{2}$$

Faster convergence properties have been claimed by optimizing the shape of the activation function. For instance, the adaption of the temperature factor of the sigmoid activation function simultaneously with the weights by means of the generalized delta learning rule can reduce the required training time significantly. However, prior knowledge about the active area of the activation function is required in order to be able to scale the input variables into this range. Also the importance of reducing the training time is sometimes overstated since the computational power of the current generation of computers is satisfactory to obtain neural network models within acceptable time.

2.3. The RBF network

In this section only a brief explanation about RBF network topology and mathematics is given. For the theoretical backgrounds the reader is referred to e.g. Refs. [6,8–10,16,17]. RBF networks consist generally of a three-layered feed-forward structure. The input layer has, like any other network, no calculation power and merely distributes the input information to the kernel functions. In this case, the parameters to be adjusted are the centroids of the kernel functions (represented by $\boldsymbol{c}_j$) and the scaling factors $\boldsymbol{\sigma}_j$. The output units are calculated by a linear combination or a weighted sum of the kernel functions, according to

$$y_i=\sum_{j=1}^{L} \boldsymbol{\lambda}_j\, \boldsymbol{\Phi}_j\left(\|\boldsymbol{x}_i-\boldsymbol{c}_j\|\right) \tag{3}$$

The most common kernel is the Gaussian radial base function, given by

$$\boldsymbol{\Phi}_j\left(\|\boldsymbol{x}_i-\boldsymbol{c}_j\|\right)=\exp\left(-\|\boldsymbol{x}_i-\boldsymbol{c}_j\|^2/\boldsymbol{\sigma}_j^2\right) \tag{4}$$

Fig. 1 can be used as a graphical representation of the RBF neural network when $\boldsymbol{W}_j$ is substituted by $\boldsymbol{c}_j$. The width factor $\boldsymbol{\sigma}_j$ represents a distance scaling parameter that determines the proportion of the input space where the jth RBF will have significant nonzero response. The location of the centroids of the kernel j is obtained by vector $\boldsymbol{c}_j$. If $\boldsymbol{\sigma}_j$ is fixed, network training yields the optimal weights $\boldsymbol{\lambda}_j$ and a vector $\boldsymbol{c}_j$. It is also possible to train the scaling parameters simultaneously or in a successive phase by means of the generalized delta learning rule [10]. If $\boldsymbol{\sigma}_j$ is fixed, prior information about the data is required since the response in input space is determined by the scaling parameters. Small values for $\boldsymbol{\sigma}_j$ exhibit local responses. However, if $\boldsymbol{\sigma}_j$ is increased, interpolation starts to occur within the range of the training data. If $\boldsymbol{\sigma}_j$ is further increased, extrapolation outside of the training data is possible.

The Euclidean distance is generally used as a distance measure for the input objects to the centroid. However, other metrics have also been successfully applied.

2.4. Robustness analysis

The goal of the robustness analysis is to find a measure for the effect of random noise in input space on the random deviations emerging on the output units. In this section, a procedure for the quantification of robustness of MLF and RBF models is discussed.

The probability density functions of the errors emerging on the output units can be estimated by Monte Carlo error sampling. For instance, when a two-layered network is considered, the errors Δx_{ik} and ΔW_{kl} are sampled from the normal distributions

$$\Delta x_{ik} \in N(0, \sigma_k), \quad \Delta W_{kl} \in N(0, \sigma_{kl}) \tag{5}$$

with zero mean and variance σ^2. Note that the errors sampled from these distributions can have both a positive or a negative sign. The index for the data patterns or objects is denoted by i, k for the input variables and l for the hidden units, in correspondence with Fig. 1.

Distributional information about input and output (pure) errors (experimental conditions, measurement noise) can be estimated by replicated experiments. It is more difficult to get distributional information about weight errors since the network does not always converge to the same minimum as a result of different initiations and local minima.

The variance of the predicted output errors can be obtained by propagating the errors through the network:

$$\Delta y_i = y_i - \psi[(x_i + \Delta x)(W + \Delta W)] \tag{6}$$

$$\Delta y_i = y_i - \psi(x_i W + \Delta x W + x_i \Delta W + \Delta x \Delta W) \tag{7}$$

In Section 1 the problems of estimating the weight errors have already been addressed. A simplification has been used by setting the weight errors to zero (omitting the last two terms in Eq. (7)) so only the errors in input space are considered. Although the results of the robustness procedure are affected by

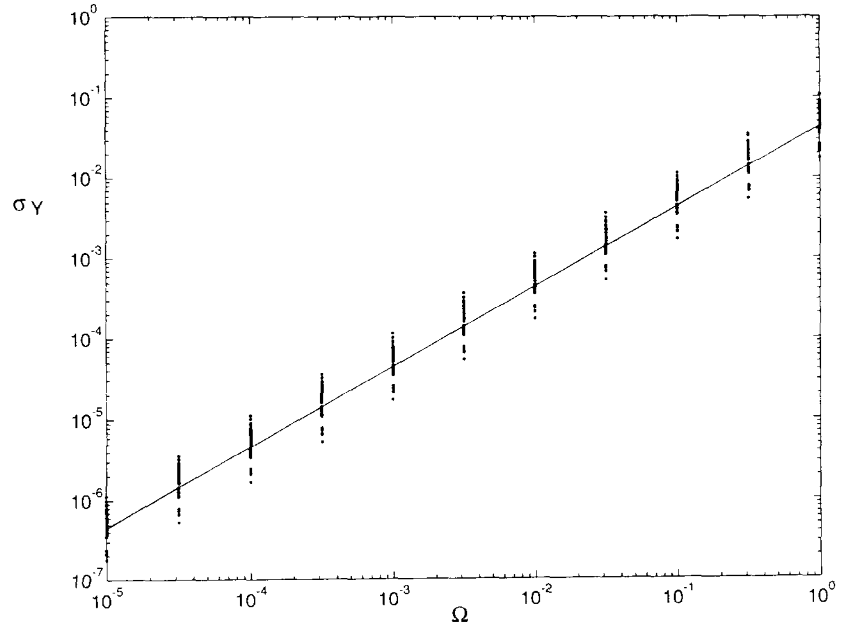

Fig. 2. Variance of an output unit as a function of the noise amplitude factor (Ω). In this example the first output unit for the Yarn data (Table 3, Section 4.2) has been used. The distribution of the objects specified by the dots is almost identical for every amplitude level, revealing that the position in input space does not severely affect the sensitivity (β_1) to input noise. Note that the offset factor (β_0) is on the right side of the figure at $\Omega = 1$ (original distribution).

omitting the model noise, the results can still be used for a qualitative judgement about the neural models.

The Monte Carlo sampling scheme can be carried out for a number of (decreasing) noise amplitudes as is shown in Fig. 2. The followed procedure is explained in more detail in Fig. 3.

The noise-amplitude factor has been used to increase the amount of noise from zero (infinite narrow distributions) to the distributions originally used ($\Omega = 1$). Since the required number of MC simulations is proportional with the dimension of the simulation space, Ω was kept smaller than one, in order to avoid a tremendous increase of computation time. For this reason the noise-amplitude factor might also be considered as a 'noise-attenuation' factor. The sensitivity of the output units to the noise-amplitude factor can be obtained by means of ordinary least squares (OLS) regression of the logarithm of the pre-

```
STEP I. INITIATION
A.      Define the input error distribution $N(0,\sigma_j)$
        for each $j$-th variable. Use prior
        information from replicated experiments.
B.      Fill in an array with attenuation factors
        ($\Omega$) in increasing order (e.g. [0..1]) in order
        to regulate the distributions

STEP II. SIMULATIONS
For Attenuation_Factor_Index  t=1 to Number_Of_Factors
        For Simulation_Run=1 to Number_Of_Runs
                1. Generate 'noisy' input patterns
                using information from A and
                amplitude factor $\Omega$ from B.
                (example:
                        $\tilde{x}=[x(1), x(2), .. , x(K)]'+\Omega[\Delta x(1), \Delta x(2), .. , \Delta x(K)]'$
                whereas $\Delta x_j \in N(0,\sigma_j)$ )
                2. Propagate the simulated patterns
                to the output of the network
        End

        Calculate the variance for each output unit at
        the noise factor $\Omega$.

End

STEP III. REGRESSION
Model the variance as a function of the noise amplitude factor.
```

Fig. 3. The three steps to quantify the sensitivity of the output variables to the amplitude of noise on input patterns.

dicted error against the logarithm of the noise-amplitude factor (Ω) given by

$$\log(\mathrm{var}(\Delta y_i)) = \beta_0 + \beta_1 \log(\Omega) \qquad (8)$$

The offset parameter β_0 gives an indication of the pure output errors, averaged over all the objects. The slope, represented by parameter β_1, contains information about the sensitivity to the amount of noise in input space.

Certain assumptions have to be made. The sensitivity of the output units can vary for each location in input space. However, the sensitivities for all the positions in input space are assumed to be equal. From Fig. 2 it can be concluded that the position in input space defined by the objects has more effect on the estimated 'pure' error than on the sensitivity. Thus, applying OLS regression, an indication about 'pure' errors and sensitivity averaged over the total input space can be obtained. This still does not give us information about the predictability of the network in terms of confidence bands which is in agreement with Ref. [18]. However, it provides a validation tool to judge the probability of false estimations when noisy input patterns are offered to the network.

3. Materials and methods

The programs required for creating RBF and MLF neural network models and the robustness analysis have been developed in Matlab 4.2™, which is a computing environment for high-performance numeric computation and visualisation. The Matlab programs have been executed on various Sparc-10™ workstations. On these computers, a training procedure for a MLF network (Yarn data, Section 4.2.1) containing nine hidden units, took about 50 s execution time, whereas the robustness analysis took approximately 30 min on a Solaris Sparc-10™ workstation. Since the RBF models contained more base functions, the execution time increased up to a factor three.

Error backpropagation using the gradient descent method [2] was used in all cases to train the network weights. Batch training was applied (i.e. propagating all data simultaneously through the network) to speed up the training and to avoid the dependency of the delta learning rule on the sequence of the input pattern vectors.

4. Experimental

Both MLF and RBF neural networks were trained for resolving a series of simulated function approximation problems from Sekulic et al. [19]. The data, generated by the additive, interactive, complex and transformed data models, have originally been used for testing the modelling capabilities of MARS (multivariate adaptive regression splines). Additional to the original smooth simulated data sets, a noisy data set was included for each type of data in order to test the generalization ability of the network models. Once the networks had been trained, the robustness analysis was applied.

Next to that, a data set from chemical practice was used for modelling the relation between physical structure and mechanical properties of yarns [20]. Since statistical information about the input and output errors were available, the predictive ability could be verified.

Among all the possibilities, both networks were configured identically with respect to the learning rates, network topology and the training method in order to make a reasonable comparison.

Besides that, the Kohonen learning rule was used to initialize the RBF network, i.e. adapting the centroids. In our case, better models were obtained and a more founded comparison with the MLF network was possible. The required number of network parameters was estimated by means of crossvalidation procedures. Parsimonious models were obtained in this way and overtraining effects were avoided.

The neural models were validated by means of the root mean squared error of prediction (RMSEP) computed only in the test data sets. When the network consists of multiple output, the overall RMSE (or RMSEP for predictions) which represents the norm of the various output units, has been used. The formulas for the prediction errors are

$$E_k = \sqrt{\sum_{p=1}^{N} \frac{\left(\hat{Y}_{pk} - Y_{pk}\right)^2}{N}} \qquad (9)$$

$$E_{\mathrm{overall}} = \sqrt{\sum_{k=1}^{M} \frac{E_k^2}{M}} \qquad (10)$$

where M represents the number of output variables.

In this paper, the network parameters of both types of networks have been adapted by means of a gradient descent method like the generalized delta learning rule. In contrast to MLF, RBF parameters are adapted in a linear fashion. As a result of the linear parameter adjustment and the concept of narrow receptive fields of the basis functions, RBF is suited for faster learning. Since training affects only a small neighbourhood of the interpolating function, previously learning paths of the function are not corrupted.

However, the required number of radial basis functions is proportional to the distribution of the data objects in input space and moreover the complexity of the data used. Especially when high-dimensional data are used, many radial base functions are required, resulting in extensive slow learning networks. Simulation studies for function approximation problems have shown the superiority of RBF to MLF when low-dimensional input spaces are used [9].

4.1. Simulated data

4.1.1. Data description

The training sets were generated by feeding a grid of 121 knots or a 11^2 factorial design over the range [0, 1] to the additive, interactive, complex and transformed data models as described by Sekulic et al. [19]. A general formula for these models is

$$y = A F(x_1, x_2) + \epsilon \tag{11}$$

where x_1 and x_2 represent the two factors of the factorial design. In accordance with Ref. [19], additional scaling parameters and an amplitude factor A are included.

Similar to the procedure outlined above, noisy data sets were generated. Here, a small amount of noise sampled from a normal distribution $\epsilon_i \in N(\mu, \sigma/100)$ was added to the smooth data sets, yielding an error ratio of approximately 1%.

Both the smooth and noisy data sets, representing the additive, interactive, complex and transformed data models, were used to compare the robustness of RBF and MLF networks.

The test sets, each containing 500 objects, were composed by uniform random sampling from input space and feeding these objects to the various data models as is illustrated for the complex data model in Fig. 4.

4.1.2. Results

The results of the MLF and RBF models are summarized in Table 1. For all the data models, the num-

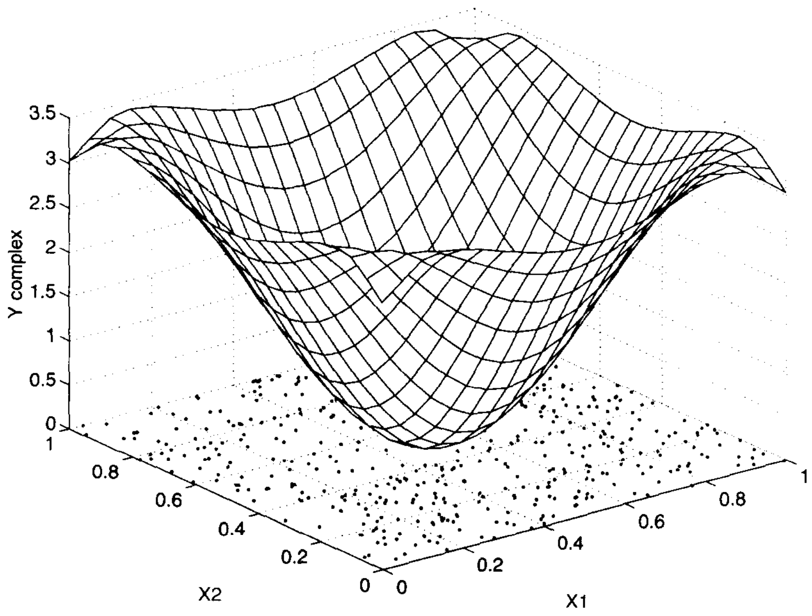

Fig. 4. The complex data model as defined by Ref. [19]. The input patterns for the test set (randomly selected in the input space) are denoted by the dots.

ber of hidden units or base functions and the root mean squared errors of calibration (RMSE) and prediction (RMSEP) are included.

The prediction errors denoted by the root mean squared error of prediction (RMSEP), as shown in Table 1, indicate that especially the highly nonlinear data are generally better modelled by the RBF network. In addition to that, the RBF models perform equally for every simulated data model. In this case, the disadvantage of the RBF models is that they need more hidden units or base functions to establish the neural model. In addition to that, it needs to be emphasized that our experience indicates that RBF networks, trained by the generalized delta learning rule, generally suffer more from convergence problems than MLF networks and more effort has to be invested to train the network.

4.2. Practice data

4.2.1. Data description

A data set from chemical practice by De Weijer et al. [20] was used for modelling the relationship between physical structure and mechanical properties of industrial poly(ethylene terephthalate) yarns. In his paper, De Weijer describes how MLF networks were applied to model the dependency of the thermal and mechanical properties to industrial process conditions.

The most important characteristics of the physical structure are the crystallinity, size and orientation of the molecules. In total, 11 structure quantities were measured. The mechanical yarn properties were described by five parameters containing information about tensile strength, energy, absorbance, elongation and modulus. For theoretical backgrounds and information about the measurements and experimental conditions, the reader is referred to Ref. [20].

The data consisted of 294 yarns. From these data, 50 test yarns were selected for evaluating the predictive ability of the neural models during training. From a crossvalidation procedure for estimating the required number of hidden units it appeared that 8 hidden units for the MLF network and 26 base functions for the RBF network yielded the best models with respect to the bias–variance trade-off.

The experimental conditions, for making the neural models, were kept identical to the original paper, i.e. the input values were scaled between −1 and 1, and the output values were scaled between 0 and 1.

4.2.2. Results

Both MLF and RBF networks were applied in order to model the physical structure and mechanical

Table 1
The root mean squared errors of calibration (RMSE) and prediction (RMSEP) for the (additive, interactive, complex, transformed and exponentially transformed) simulated data. The optimal number of hidden units (HU), estimated by crossvalidation, is also specified

Simulated data						
Data model	MLF			RBF		
	HU	RMSE	RMSEP	HU	RMSE	RMSEP
Additive						
Smooth	7	0.0328	0.0233	8	0.0352	0.0304
Noise	7	0.0337	0.0244	10	0.0297	0.0250
Interactive						
Smooth	6	0.0886	0.0774	10	0.0287	0.0235
Noise	6	0.0428	0.0371	9	0.0346	0.0294
Complex						
Smooth	7	0.2014	0.1882	10	0.0536	0.0384
Noise	7	0.1994	0.1815	10	0.0588	0.0431
Transformed						
Smooth	7	0.0241	0.0190	10	0.0455	0.0368
Noise	7	0.0206	0.0213	10	0.0372	0.0310
Exponentially transformed						
Smooth	5	0.0368	0.0321	9	0.0385	0.0319
Noise	4	0.0458	0.0408	10	0.0372	0.0310

Table 2
The percentage explained variance (PEV) and the root mean squared error of prediction (RMSEP) for all the output variables of the MLF and RBF network models (Yarn data)

Practice data

Property	MLF model		RBF model	
	PEV (%)	RMSEP	PEV (%)	RMSEP
1	95.85	0.0828	94.88	0.0920
2	65.35	0.2234	66.82	0.2186
3	96.39	0.1192	96.10	0.1239
4	92.76	0.1226	93.59	0.1153
5	92.77	0.1415	93.13	0.1380
6	92.33	0.1569	92.89	0.1511
7	97.37	0.0701	96.32	0.0829
8	93.04	0.1057	92.66	0.1085
9	78.83	0.2422	79.73	0.2370
10	93.39	0.1241	93.52	0.1229
11	59.38	0.2179	59.08	0.2187

properties of industrial yarns, from the paper of De Weijer et al. [20]. In Table 2 the percentage of explained variance (PEV) and the RMSEP values for the eleven mechanical properties are presented. From crossvalidation it appeared that the MLF models required 8 hidden units, whereas the RBF model needed up to 26 hidden units (base functions) to reach a comparable degree of accuracy. In agreement with the previous results, the MLF network is better capable to create parsimonious models with respect to the RMSE and the size of the network.

However, Table 3 reveals some interesting properties of the RBF network. The output units of the RBF model are generally more robust to input noise for most of the mechanical properties, which is in accordance with Ref. [10]. Also a better estimation of the 'pure' output errors can be observed.

5. Discussion and conclusion

In this paper, we have proposed a method to test the robustness properties of two different types of neural networks (radial base function (RBF) and multi-layered feed-forward (MLF) networks) applied to a data set obtained from industrial chemical practice. The method can be used as a qualitative validation tool to investigate the sensitivity of network outputs to the amplitude of random errors in input space. Since the results of the robustness analysis depend on the data and the network configuration used, no general information can be extracted about intrinsic robustness properties. Besides that, prior knowledge about pure errors in input and output space is required. In the experimental section of this paper, the robustness procedure has been applied on Yarn data [20] modelled by RBF and MLF neural networks.

An appealing feature of RBF networks is that the network parameters have physical meanings in contrast to MLF networks. The weights connected be-

Table 3
The results of the robustness analysis (Yarn data). The original pure errors for the 11 output variables are included in order to compare them with the offset coefficients (β_0) of the MLF and RBF neural models. The sensitivity of the output units to the amplitude of input noise is denoted by β_1

Practice data

Property	Pure error	MLF model		RBF model	
		β_0	β_1	β_0	β_1
1	0.0472	0.0524	1.0011	0.0579	1.0007
2	0.1596	0.0810	1.0011	0.1032	0.9999
3	0.0927	0.0476	1.0043	0.0830	1.0011
4	0.0918	0.0512	1.0003	0.0550	0.9996
5	0.1064	0.0539	1.0000	0.0537	1.0010
6	0.1370	0.0444	1.0049	0.0897	1.0018
7	0.0798	0.0439	1.0032	0.0626	1.0007
8	0.0808	0.0644	1.0010	0.0677	1.0003
9	0.1100	0.0918	1.0009	0.1054	0.9993
10	–	0.0533	1.0002	0.0548	1.0007
11	0.3276	0.0467	1.0023	0.0539	1.0019

tween the base function and the variable represent the coordinates of the centre of a group of objects. The output weights are used to realize a weighted sum of center positions. In this way, a specific subdomain in the data space can be selected. The scaling parameter σ is used to adapt the narrowness of the Gaussian-shaped functions.

The RBF network falls between a nearest neighbour matcher and a regression technique. The scaling parameter σ can be used to regulate the behaviour of the network. Narrow Gaussian distributions make it possible to approximate linear or nonlinear correlations in an N-dimensional space. It is obvious that the RBF network needs much more base functions for function approximation than for classifications problems. When the network is applied for classification, higher values for σ are more common. The radial base functions are then used as centroids for a group of objects in a hyperspace. Note that one cluster can be distributed over various base functions. A linear combination of base functions pointing to the same class enables a better discrimination between different classes.

For simulated function approximation problems used in this paper, MLF outperforms RBF, i.e. parsimonious models are obtained with lower prediction errors (Table 1). However, both types of networks still remain comparable.

On the yarn data, a robustness analysis has been carried out. Table 3 shows that the RBF model is generally more robust for input noise. Also, the intercept gives a better approximation of the pure errors. This allows us to conclude that the RBF model has better robustness properties than MLF models.

The study includes a restricted comparison of the speed and convergence properties for both types of networks. With the simulated data used for function approximation in various models, linear and nonlinear, using crossvalidation procedures to avoid the overtraining effect, there are no big differences between MLF and RBF networks. Both converge approximately in the same range, but MLF models were computationally less expensive than RBF, which however is not a big problem taking into account the power of modern computers.

On the other hand, the RBF models for yarn data appeared to be more robust than the MLF networks obtained. Summarizing, the choice of a particular type of network depends on the problem one tries to resolve, the available computational power and the goal pursued.

Acknowledgements

The authors wish to acknowledge A.P. de Weijer for handing out background information and valuable suggestions about the yarn data. A. Bos and W. Melssen are acknowledged for the fruitful discussions about neural network theory.

This research work is financially supported by the Dutch Foundation for Chemical Research (SON) on behalf of the Dutch Organisation for Scientific Research (NWO).

References

[1] P.K. Simpson, Artificial Neural Systems, Pergamon Press, 1990.

[2] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Vol 1. Foundations, MIT Press, Cambridge, MA, 1986.

[3] H.C. Smit, Specification and estimation of noisy analytical signals. Part I. Characterization, time invariant filtering and signal approximation, Chemom. Intell. Lab. Syst., 8 (1990) 15–27.

[4] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley, Reading, MA, 1990.

[5] K. Hornik, M. Stinchcombe and H. White, Multilayered feed-forward networks are universal approximators, Neural Networks, 2 (1089) 359–366.

[6] J. Moody and C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation, 1 (1989) 281–294.

[7] P. Cardaliaguet and G. Euvrard, Approximation of a function and its derivative with a neural network, Neural Networks, 5 (1992) 207–220.

[8] J. Park and I.W. Sandberg, Universal approximation using radial basis function networks, Neural Computation, 3 (1991) 246–257.

[9] D.A. White and D.A. Sofge, Handbook of Intelligent Control, Van Nostrand Reinhold, New York, 1992.

[10] A. Bos, Artificial neural networks as a tool in chemometrics, PhD thesis, University Twente, Enschede, 1993.

[11] S. Kirkpatrick, J. Gelatt and M.P. Vecchi, Optimization by simulated annealing, Science, 220 (1983) 671– 680.

[12] D. Burshtein, Neural network training using the simulated annealing method, in Proceedings of the World Congress on Neural Networks, Vol. 3, Lawrence Erlbaum, Hillsdale, 1993, pp. 397–400.

[13] J.D. Schaffer, D. Whitley and L.J. Eshelman, Combinations of genetic algorithms and neural networks: A survey of the state of the art, in the Proceedings of COGANN-92, IEEE Computer Society, Los Alamitos, CA, 1992, pp. 1–13.

[14] I. Youji and H. Sakai, real-time learning algorithm for a multilayered neural network based on the extended Kalman filter, IEEE Trans. Signal Process., 40 (1992) 959–966.

[15] J.R.M. Smits, Exploring the possibilities of applying artificial neural networks on problems in analytical chemistry, PhD thesis, Catholic University Nijmegen, Nijmegen, 1993.

[16] D.S. Broomhead and D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Syst., 2 (1988) 321–355.

[17] M.J.D. Powell, Radial base functions for multivariable interpolation, a review, presented at the IMA Conference, RMCS, Shrivenham, 1985.

[18] J.A. Leonard, M.A. Kramer and L.H. Ungar, A neural network architecture that computes its own reliability, Comput. Chem. Eng., 16 (1992) 819–836.

[19] S. Sekulic and B. Kowalski, Mars: a tutorial, Journal of Chemometrics, 4 (1992) 199–216.

[20] A.P. de Weijer, L. Buydens, G. Kateman and H.M. Heuvel, Neural networks used as a soft-modelling technique for quantitative description of the inner relation between physical structure and mechanical properties of poly(ethylene terephthalate) yarns, Chemom. Intell. Lab. Syst., 16 (1992) 77–86.