A NEURAL NETWORK ARCHITECTURE THAT COMPUTES ITS OWN RELIABILITY

J. A. LEONARD,[1] M. A. KRAMER[1] and L. H. UNGAR[2]

[1]Laboratory for Intelligent Systems in Process Engineering, Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

[2]Department of Chemical Engineering, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.

(Received 23 August 1991; final revision received 10 February 1992; received for publication 27 February 1992)

Abstract—Artificial neural networks (ANNs) have been used to construct empirical nonlinear models of process data. Because network models are not based on physical theory and contain nonlinearities, their predictions are suspect when extrapolating beyond the range of the original training data. With multiple correlated inputs, it is difficult to recognize when the network is extrapolating. Furthermore, due to non-uniform distribution of the training examples and noise over the domain, the network may have local areas of poor fit even when not extrapolating. Standard measures of network performance give no indication of regions of locally poor fit or possible errors due to extrapolation. This paper introduces the "validity index network" (VI-net), an extension of radial basis function networks (RBFN), that calculates the reliability and the confidence of its output and indicates local regions of poor fit and extrapolation. Because RBFNs use a composition of local fits to the data, they are readily adapted to predict local fitting accuracy. The VI-net can also detect novel input patterns in classification problems, provided that the inputs to the classifier are real values. The reliability measures of the VI-net are implemented as additional output nodes of the underlying RBFN. Weights associated with the reliability nodes are given analytically based on training statistics from the fitting of the target function, and thus the reliability measures can be added to a standard RBFN with no additional training effort.

1. INTRODUCTION

Neural networks have attracted much interest lately for their use as predictive models as well as for pattern recognition. Neural networks have been used successfully to model dynamic and nonlinear systems such as deterministic chaos (Lapedes and Farber, 1987; Ydstie, 1990; Levin, 1990), chaotic chemical systems (Admoaitis *et al.*, 1989) and other chemical reactions (Bhat *et al.*, 1990). Neural networks have been used for system identification and control by many researchers (Donat *et al.*, 1990; Hernandez and Arkun, 1990; Narendra and Parthasarathy, 1990; Psichogios and Ungar, 1991; Haesloop and Holt, 1990; Willis *et al.*, 1991) and applied to process fault diagnosis by Hoskins and Himmelblau (1988), Watanabe *et al.* (1989), Venkatasubramanian *et al.* (1990) and others. Neural networks can perform computations quickly and efficiently even when the dimensionality of the problem is large because of their highly parallel structure. Neural networks also have a powerful representational capacity. Cybenko (1989) has shown that a backpropagation network (BPN) with one hidden layer is sufficient to approximate any function given enough nodes in the hidden layer. The radial basis function network (RBFN) has been shown to have a similar capability to represent arbitrary functions (Park and Sandberg, 1991).

The reliability of a neural network model (or any other empirical model) is determined by two factors: whether or not the model is being applied in a domain of the independent variables where training data were available (extrapolation), and the accuracy of the model for given values of the independent variables (local goodness of fit). Measures of extrapolation and local goodness of fit have not been defined in the context of neural network models. Because networks do not automatically flag when they are extrapolating from the training data, users may inadvertently use an invalid prediction. In addition, predictions of conventional networks do not carry explicit error bars, so the user has only a rough idea of the accuracy of a given prediction. This paper introduces a novel network, the validity index network (VI-net), based on radial basis function networks (RBFNs), that includes additional outputs indicating extrapolation and confidence limits on network predictions.

The simplest treatment of extrapolation involves placing individual upper and lower bounds on the permissible range of each independent variable. This approach will frequently overestimate the region of validity of the model since it assumes the independent

variables are uncorrelated. Correlations among the independent variables can be accounted for by the Mahalanobis distance (Duda and Hart, 1973), which measures the distance from a test point to the centroid of the training data scaled by the variance in the direction to the centroid. Strictly, the Mahalanobis distance is an appropriate measure of extrapolation only if the independent variables follow a multivariate Gaussian distribution. Another possible definition of extrapolation in multiple dimensions is based on the convex hull of the training data. The convex hull is the minimum volume polyhedron circumscribing the training points (Chand and Kapur, 1978). However, if there are nonconvexities, training data may be absent from regions inside the convex hull.

Instead of assuming a specific distribution or convexity, the VI-net implements a nonparametric, nonconvex method of determining when there are sufficient training data for a reliable model prediction. Extrapolation in this context is defined as any local region of the input space with little or no training data to support a model prediction. Extrapolation is thus linked to the estimation of local training data *density* and a warning is generated when the local density of training points falls below a certain threshold. In pattern classification applications, the same indicator can be used to detect novel cases not adequately represented by the training data, provided the inputs to the classifier are real-valued.

The second factor affecting the reliability is the accuracy when the network is not extrapolating. Conventional techniques for training neural networks use only *global* measures of goodness of fit, usually in terms of the training error, error on a test set or prediction sum of squares error. However, local regions of poor fit may not be strongly reflected in the global measures or prediction error. A *local* goodness of fit can be expressed as a confidence interval on each network output, a function of the independent variables. A separate goodness of fit measure is needed for each output, since different dependent variables may be fit with different accuracy. Therefore, a neural network model with (nominally) M outputs will require $M + 1$ additional output nodes to indicate the reliability of the model predictions: M estimates of the accuracy of each model prediction, and one output to indicate extrapolation, as shown in Fig. 1.

Clearly in the discussion of a "local" error measure, the definition of "local" is crucial. If the neighborhood considered to be local to the test point is too small, it will not contain enough data to estimate the model accuracy and the test for extrapolation/lack of data will be overly sensitive. On the other hand, if the neighborhood is too large, the local variations in accuracy will be missed and the test for extrapolation/lack of data will be insensitive. In the extreme case of one large neighborhood, the global results, i.e. a global goodness of fit and no ability to detect extrapolation, should be regained. While in general the definition of neighborhood size appears to be a somewhat arbitrary feature of this approach, for RBFNs the concept of a local neighborhood is naturally well defined. RBFNs partition the training data into local clusters and limit the distance over which data may influence the model prediction. RBFN outputs are essentially a composition of multiple, overlapping, local fits to the training data. Therefore, RBFNs provide the basic framework for development of the VI-net.

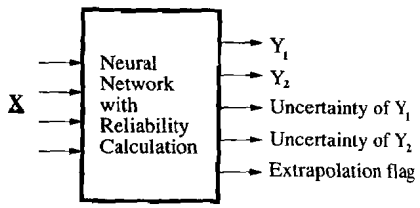

Fig. 1. Neural network model with additional outputs representing local reliability.

This paper is organized as follows. Section 2 presents a brief summary of RBFNs, including the method of cross validation used to estimate the expected error of the model and to determine the appropriate model structure. Section 3 presents the specific definitions of the different reliability measures as they are implemented in the VI-net. These concepts are extended to pattern classification in Section 4. Three examples of the application of the VI-net are presented in Section 5. Section 6 summarizes this work and concludes with a discussion of why similar reliability measures would be difficult to implement within the BPN framework.

2. RADIAL BASIS FUNCTION NETWORKS

Several authors have studied the use of RBFs (Medgassy, 1961) as transfer functions for neural networks instead of the sigmoid function typically used in BPNs. Poggio and Girosi (1990) compared the theoretical properties of RBFNs and other networks. Moody and Darken (1989) introduced an improved training procedure for RBFNs. RBFNs have been used for many applications including phoneme classification (Renals and Rohwer, 1989), fault diagnosis problems in chemical processes (Yao and Zafiriou, 1990; Leonard and Kramer, 1991) and process control (Parasarathy and Narendra, 1991). Unlike BPNs, RBFNs use a distance metric in the

input space to determine the hidden layer activations. As a result, the contours of constant activation of the hidden layer are hyperspheres instead of hyperplanes as with BPNs. The contours are finite in length and form closed regions of significant activation, as opposed to BPNs where the contours are infinite in length and form semi-infinite regions of significant activation. This feature of RBFNs is exploited to produce local reliability measures in the VI-net.

Radial basis function networks consist of three layers of nodes with successive layers exhaustively interconnected by feedforward arcs (Fig. 2). The first layer connections implement a fanout of the inputs to the hidden layer, and these connections are *not* weighted as in BPNs and the input to each hidden node is the input vector **x**. The hidden layer consists of H radial units plus a bias node which has a constant activation of one. The transfer functions in the hidden nodes are similar to the multivariate Gaussian density function:

$$a_h = \exp(-\|\mathbf{x} - \boldsymbol{x}_h\|^2/\sigma_h^2) \tag{1}$$

where a_h is the output of unit h in the hidden layer given the input **x**. Each RBF node has $N + 1$ internal parameters: $\boldsymbol{x}_h$, the position of the center of the radial unit in the feature space and σ, a distance scaling parameter known as the unit width which determines over what distance in the input space the unit will have a significant influence. The parameter σ has the same function as the standard deviation in the standard normal probability distribution, although it is not estimated in the same way. Note that unlike the sigmoid function used in BPNs, the transfer function in the RBF nodes is nonmonotonic. Each RBF unit has a significant activation over a specific region determined by $\boldsymbol{x}_h$ and σ_h, thus each RBF unit represents a unique local neighborhood in the input space.

The connections in the second layer of the RBFN are weighted in the usual neural network manner and the output nodes of the RBFN are linear summation units. The value of the ith output node y_i is given by:

$$y_i = \sum_{h=1}^{H+1} w_{ih} a_h. \tag{2}$$

The bias node is represented as $a_{H+1} = 1$.

The method of Moody and Darken (1989) trains RBFNs in three sequential stages, as opposed to the single optimization procedure used in BPN training. The first stage consists of determining the unit centers $\boldsymbol{x}_h$ by the k-means clustering algorithm (MacQueen, 1967), an unsupervised technique that places unit centers centrally among clusters of training points. Next the unit widths are determined using a nearest

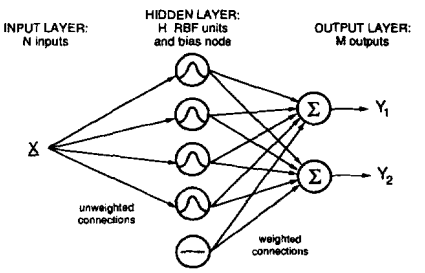

Fig. 2. Radial basis function network (RBFN) architecture.

neighbor heuristic that ensures the smoothness and continuity of the fitted function. The width of any hidden unit is taken as the RMS (square root of the mean square) distance to the P nearest unit centers, where P is a design parameter. Finally the weights of the second layer of connections are determined by linear regression. The objective function to be minimized is the RMS error between network output and training targets, the same as for BPNs.

Each group of parameters for the radial basis function network can be trained in polynomial time. In general, the training of a RBFN is an order of magnitude faster than the training of a comparably-sized BPN, even when using efficient versions of backpropagation.

Several variations on RBFN architecture and training have been proposed (Holcomb and Morari, 1991; Jokinen, 1991; Sanger, 1991). Variations include replacing the RBF hyperspheres with hyperellipses, optimizing unit centers and/or widths by backpropagation training, using sequential allocation of unit centers, etc. In this paper, we limit attention to the network architecture and training method just described. However, the concepts for reliability prediction in the VI-net may be extendable to other variations on RBFNs.

2.1. *Determining model structure*

In addition to determining the values of the network parameters, one must also select the appropriate network structure. In the case of RBFNs this means choosing the number of RBF units and the value of the overlap parameter P for the nearest neighbor heuristic. Since there is usually a diminishing returns effect on addition of hidden nodes, one method of selecting the number of hidden nodes is to choose a network near the "knee" on the curve of training error vs number of hidden nodes. Ljung (1987) suggests selecting the network that minimizes Akaike's information theoretic criteria (AIC), which makes explicit the trade-off between the improved fit to the training data achieved by adding more parameters to the model and the added complexity of this bigger model. Still another method of selecting an

optimal hidden node architecture is to split the training data into two subsets. The network is trained on part of the data while the other part is held back to check on the goodness of fit. The train/test procedure is relevant here because network performance on the test set is an unbiased estimate of the expected error rate of the network on future data, which is one of the desired reliability measures. However, the disadvantage is that the network is trained on only a fraction of the available data.

The train/test method is a simplified version of the general procedure known as cross validation. The S-fold cross validation procedure (SFCV), based on statistical resampling theory, makes maximum use of the available data while still giving an unbiased error estimate (Weiss and Kulikowski, 1991). For this method, the training data are randomly partitioned into S equally-sized subsets. Typically S ranges from 10 to K, where K is the total number of training examples. When $S = K$, this procedure becomes the "leave-one-out" procedure. Weiss and Kulikowski recommended using $S = K$ for $K < 100$ and $S = 10$ for $K \geqslant 100$. For a given model structure, S different networks are trained using all of the data *except* one of the S subsets. Following training, the error is determined for each example in the subset that was held out. Each example is used to train all the networks except one, and each example has an error associated with it based on the network that did not use that example for training. If $S \geqslant 10$ then each of the S networks will have seen at least 90% of the training data and therefore should perform similarly to a network of the same structure trained on all of the training data. The mean square testing error over all the examples can be shown to be an unbiased estimate of the application error. The optimal network architecture is the one with minimum estimated application error rate.

In practice, the k-means clustering algorithm can yield different clusterings for different initial conditions for the clustering algorithm. Ideally, one should repeat the entire S-fold procedure several times to determine the mean and variation of the unbiased error estimates for each architecture. Once the optimal P and H have been found, the parameters for the working network (used as the basis for the VI-net) are determined using all the available data.

3. LOCAL RELIABILITY MEASURES

As discussed earlier, determining the reliability of any model output has two parts. First, one must determine whether or not the model is extrapolating. If so, then the model output is unreliable because the network has not learned what output is associated with the given input. If the network is not extrapolating, then one needs to estimate the accuracy of the model fit, which is given as a confidence interval on each network output. The implementation of each of these measures is presented in detail below.

3.1. Extrapolation/data density measure

The goal of the extrapolation measure is to determine if there is enough training data in the vicinity of the test point to reliably make a prediction. We have studied two different extrapolation measures, the first being the maximum activation of the RBF units:

$$\text{max-act} = \max[a_h(\mathbf{x})], \quad 1 \leqslant h \leqslant H, \tag{3}$$

max-act has a simple conceptual basis. Since each RBF unit is "centered" on a subset of the training data by the k-means clustering algorithm, as the test point x moves away from the training data, the value of the maximum activation will decrease. A small value for the maximum RBF unit activation, e.g. < 0.5, indicates extrapolation. While this is the most obvious definition for an extrapolation measure, max-act only indicates that *some* training data are in the vicinity of the test point, not the amount or density of training data in the region. Since it does not reflect data sparsity, max-act is not an ideal measure of the sufficiency of the training data. A more satisfactory measure is the local probability density of training data.

A well-known method of density estimation when the form of the probability density function is not known *a priori* is Parzen windows (Parzen, 1962). The Parzen estimator is given by:

$$\rho(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\sigma^N} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_k}{\sigma}\right), \tag{4}$$

where K is the number of training points, N is the dimension of the input and φ is a window function satisfying:

$$\int_{-\infty}^{\infty} \varphi(\mathbf{y})\mathrm{d}\mathbf{y} = 1,$$

$$\varphi(\mathbf{y}) \geqslant 0.$$

Specht (1990) has shown how Parzen windows can be implemented by radial basis function networks. Direct implementation of Parzen windows in network form requires radial units of identical width centered at every data point. However, in RBFNs considered here, there are fewer units than data points, radial units are not centered at data points and unit widths are not identical. Tråvén (1991) also uses radial units to estimate probability densities, but the connection weights, unit centers and unit widths are optimized for the density estimation task. We would like to

estimate the data density using the unit centers and widths determined when the RBFN is trained for functional approximation.

Instead of estimating the density at a test point by direct application of Parzen windows, a two-stage approach is used. First, densities are estimated at each hidden unit center using Parzen windows. Then, based on these densities, the densities at arbitrary test points are determined by means of an interpolation formula. At hidden unit h, the Parzen density estimate based on the unit width σ_h and unit activation function $a_h(x)$ is:

$$\rho_h = \rho(x_h) = \frac{1}{KV_h} \sum_{k=1}^{K} a_h(\mathbf{x}_k),$$

where

$$V_h = \int_{-\infty}^{\infty} a_h(\mathbf{x})\, d\mathbf{x}. \tag{5}$$

To derive an appropriate interpolation formula for $\rho(\mathbf{x})$ given the densities at the unit centers, we introduce a simplified model where the activation functions are sharply-defined hyperspheres, rather than Gaussian distributions. Assume the hidden node activations assume binary values based on the distance from the unit centers:

$$\begin{aligned} a_h(\mathbf{x}) &= 1 \text{ if } \|\mathbf{x} - \mathbf{x}_h\| \leqslant c \cdot \sigma_h \\ &= 0 \text{ if } \|\mathbf{x} - \mathbf{x}_h\| > c \cdot \sigma_h. \end{aligned} \tag{6}$$

The constant c is defined as the smallest positive real number that allows the hyperspheres to cover the training data completely. Since k-means clustering places units in clusters of training points and the P nearest neighbor heuristic chooses unit widths according to the distance between centers, c will be an order one constant, but the exact value is unimportant. Given this activation function, equation (5) simplifies to (Duda and Hart, 1973):

$$\rho_h = (n_h/K)/V_h \tag{7}$$

where V_h is the volume of hypersphere h, n_h is the number of training points falling inside V_h and K is the total number of training points. Note that because the hidden units overlap, $\Sigma_h n_h \neq K$.

Because an arbitrary point $\mathbf{x}$ may be contained in more than one hypersphere, it may be associated with more than one density estimate. An estimate of the local density for any point contained in more than one unit can be obtained by averaging the density estimates associated with all hyperspheres containing the point:

$$\rho(\mathbf{x}) = \frac{\sum_{h=1}^{H} a_h(\mathbf{x})\rho_h}{\sum_{h=1}^{H} a_h(\mathbf{x})}. \tag{8}$$

Since a_h in the simplified model assumes only the values 0 and 1, the denominator is an integer count of the number of units containing the test point. This formula becomes indeterminate if the test point $\mathbf{x}$ is contained in none of the hyperspheres. However, because all the training data is covered by the hyperspheres, if numerator and denominator are both zero, it is known that $\rho(\mathbf{x}) = 0$. Equation (8) can be generalized to include the case of a test point falling outside all hyperspheres by defining V_0 to be the complement of the space covered by the hyperspheres. The activation function for V_0 is $a_0(\mathbf{x}) = 1 - \max(a_h)$, that is, $a_0 = 1$ if and only if the activation of all hyperspheres is zero. With this definition, equation (8) can be rewritten as:

$$\begin{aligned} \rho(\mathbf{x}) &= \frac{\sum_{h=0}^{H} a_h(\mathbf{x})\rho_h}{\sum_{h=0}^{H} a_h(\mathbf{x})} \\ &= \frac{\sum_{h=1}^{H} a_h(\mathbf{x})\rho_h}{\sum_{h=1}^{H} a_h(\mathbf{x}) + 1 - \max(a_h)}, \end{aligned} \tag{9}$$

the latter equality stemming from the fact that $\rho_0(\mathbf{x}) = 0$.

Consider now the case where a_h is a continuous variable, as given by equation (1). The Parzen window estimate of the density at the unit centers, using a Gaussian window of width σ_h is:

$$\rho_h = \frac{\sum_{k=1}^{K} a_h(\mathbf{x}_k)}{K(\pi^{1/2}\sigma_h)^N}. \tag{10}$$

Note that in this case, the effective number of points associated with any hidden unit h is:

$$n_h = \sum_{k=1}^{K} a_h(\mathbf{x}_k). \tag{11}$$

To estimate the density at a novel point $\mathbf{x}$, the density contributions from active hidden units must be combined, analogous to equation (9). Although derived for crisp activation functions, equation (9) can be applied directly to continuous activation functions. With continuous activations, equation (9) calculates a *weighted* average of unit densities, rather than a simple average. The densities at the unit centers are weighted in proportion to the activation caused at each hidden unit by the test point. If the test point is close to a unit center, its activation will be high and the corresponding unit density will appear in the interpolation formula with a weight approaching one. If the test point is far away from a unit, its activation and hence its weighted contribution will be small. If

the test point is far from *all* units, then the complementary activation function will approach unity, while the numerator simultaneously approaches zero, and the resulting density will approach zero.

The use of equation (9) with continuous activation functions can also be given an interpretation based on fuzzy logic. If $a_h(\mathbf{x})$ is treated as the membership function of $\mathbf{x}$ in unit h, then the membership of $\mathbf{x}$ in the space not covered by RBF units is the fuzzy complement (Negoita, 1985) of $(a_1 \cup a_2 \cup \ldots \cup a_h)$, or $a_0 = 1 - \max(a_h)$, precisely as in the case of crisp units. Thus the intuitive extrapolation measure max-act can be interpreted as the continuous membership function of $\mathbf{x}$ in the space covered by RBF units. The contour max-act = 0.5 represents equal membership in the spaces inside and outside the space covered by RBF units.

The data density index is implemented in the VI-net by adding several auxiliary nodes to the basic RBFN architecture as shown in Fig. 3. A summation node connected with unit weights to the hidden layer calculates the denominator of equation (9), and another summation node with weights ρ_h calculates the numerator. In addition, a "maximum" node connected to the hidden layer with unit weights is included, and finally a ratio node outputs $\rho(\mathbf{x})$.

While equation (9) represents one possible way to interpolate the point estimates of unit densities (and the approach used here), other options are conceivable. Equation (10) yields a set of H points with corresponding density estimates, (x_h, ρ_h), $h = 1, \ldots, H$. It is possible to obtain an estimate of $\rho(\mathbf{x})$ passing through these points by linear regression using the existing radial basis function. In other words, we seek the parameters w_h such that $\rho(\mathbf{x}) = \Sigma_h w_h a_h(\mathbf{x})$. Defining A_H to be the square, symmetric matrix consisting of the center-to-center activations $a_h(x_j)$, then $\boldsymbol{\rho}_h = A_H \mathbf{w}_h$ and hence $\mathbf{w}_h = (A_H)^{-1}\boldsymbol{\rho}_h$. This interpretation also gives rise to a simple network implementation, a linear layer attached to the hidden nodes. The disadvantage of this approach is that the linear regression may be ill-conditioned, and because the function is forced through the density estimates at each center, a "bumpy" estimate of $\rho(\mathbf{x})$ may result. However, since $a_h(\mathbf{x})$ approaches zero as the network extrapolates, the density displays the correct asymptotic behavior.

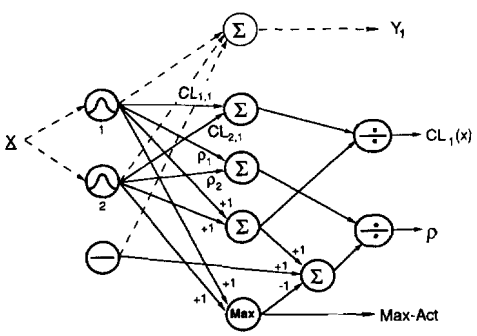

Fig. 3. The validity index network (VI-net) with reliability measures confidence limit (CL), local data density (ρ) and membership function in hidden units (max-act). The underlying RBFN is indicated with dashed lines. The case of a single dependent variable ($M = 1$) is shown, with two radial hidden units ($H = 2$).

In order to interpret $\rho(\mathbf{x})$ in terms of an "adequate amount" of data, one must set a threshold value. While this is somewhat problem dependent, a heuristic that has given good results in our experience is to set the threshold for ρ equal to the minimum value of ρ over the set of training examples, provided the confidence interval local to the minimum ρ is acceptable. If the model accuracy is less than acceptable at the minimum ρ, then a higher threshold value should be selected to correspond to an acceptable confidence interval.

3.2. *Reliability based on dependent variables*

The method proposed to model the accuracy of the fit for a given output in the region of the test point is based on analogy to the confidence limits placed on a random variable. A local confidence interval is developed for each RBF unit and then the confidence limits for the model prediction at a given test point are formed by taking a weighted average of the confidence limits over all contributing units.

Assuming momentarily that training points can be associated unambiguously with hidden units as in the crisp hypersphere model, the local estimate for the variance of the model residual in output i within the domain of hidden unit h is:

$$s_{hi}^2 = \left[\sum_{j=1}^{n_h} E_{ij}^2\right] \Big/ (n_h - 1), \tag{12}$$

where the summation is taken over the n_h points inside the hypersphere. E_{ij} is the cross-validation error on output i for training example j for the architecture of the working network, stored during the cross-validation procedure. To generalize equation (12) to Gaussian radial basis function units, we again use $a_h(\mathbf{x}_k)$ to represent the membership of training point $\mathbf{x}_k$ in unit h. Each training point is allowed to contribute to the error estimate in proportion to its membership in the unit, as in equation (9). The resulting local variance of the model residuals for output i at hidden node h is:

$$s_{hi}^2 = \left[\sum_{k=1}^{K} a_h(\mathbf{x}_k)^* E_{ik}^2\right] \Big/ (n_h - 1), \tag{13}$$

where n_h is now defined as in equation (11). The 95% confidence limit for the expected value of the residual associated with output i for unit h is given by:

$$CL_{hi} = t_{95} * s_{hi} * n_h^{-0.5}, \quad (14)$$

where t_{95} is the critical value of the Student's t-statistic for 95% confidence with $n_h - 1$ degrees of freedom. Because the effective number of data points in a radial unit n_h is a continuous-valued function, the value of t_{95} is calculated from a continuous polynomial fit of t_{95} vs $1/(n_h - 1)$. Finally, the confidence limit for a test point is an average of the RBF unit confidence limits, weighted by the contribution of each hidden unit to the output for the current input:

$$CL_i(\mathbf{x}) = \left[\sum_{h=1}^{H} a_h(\mathbf{x}) * CL_{hi}\right] \Big/ \left[\sum_{h=1}^{H} a_h(\mathbf{x})\right], \quad (15)$$

where $CL_i(\mathbf{x})$ is the confidence limit for the output i at test point $\mathbf{x}$. For test points far from any unit center, all of the activations will be low. By dividing by the sum of the activations, the value $CL_i(\mathbf{x})$ asymptotically approaches the value of CL_{hi} for the unit that is "nearest" to the test point, where "nearest" is in terms of the distance scaled by each of the units' receptive widths σ_h.

$CL_i(\mathbf{x})$ is implemented in the VI-net by taking the ratio of two summation nodes as shown in Fig. 3. The connections to one node have weights CL_{hi} so that the node calculates the numerator of equation (15). A second node, common to all $CL_i(\mathbf{x})$, represents the denominator in equation (15) (the same node used in the extrapolation measure). $CL_i(\mathbf{x})$ is then calculated by taking the ratio of these two nodes.

This method assumes the residuals of the model are independent, normally distributed with a mean of zero and a constant variance over the neighborhood defined by each RBF unit but varying from unit to unit. For the residuals to be normally distributed independent random variables, all of the systematic variation of the dependent variable has to be accounted for by the model. This implies that the model form exactly matches the form of the true function we are trying to estimate. Because the true functional form is unknown and the model is empirical, the assumption of normally distributed errors may not be completely accurate. Nonetheless, the method still provides an indication of the relative accuracy of the model predictions for the cases we have investigated.

Instead of asking for a local confidence limit of the expected value of the model prediction, one could pose the question of the limits of the dispersion of the individuals, i.e. the range of possible values for a single sample, rather than the dispersion of the mean. This can be estimated in much the same way as the confidence limit of the expected value of the model prediction. To do this, the variance of the residuals is calculated as in equation (13). Assuming the individuals are normally distributed, confidence limits on the individuals can be set based on the standard normal distribution. However, we do not know the true mean or variance of the residuals but have only a finite sample to estimate them. The mean may be nonzero because of mismatch between the empirical and true models. To account for this uncertainty in the distribution of the errors, the variance of the mean and the individuals are pooled. In deriving CL_{hi} we assumed the variance of the mean was equal to s_{hi}^2/n_h, therefore the sum of the variances is $(1 + 1/n_h)\ s_{hi}^2$ and the confidence limit for individual values of output i for unit h, CL'_{hi}, is given by:

$$CL'_{hi} = t_{95} * s_{hi} * (1 + 1/n_h)^{0.5}, \quad (16)$$

where all the terms are defined as in equation (13). The confidence limit of individuals for model prediction i is analogous to equation (14) except that CL'_{hi} is substituted for CL_{hi}. In the examples CL'_{hi} is shown as a band around the model predictions.

To summarize the method:

1. Choose a value of H and P for the RBFN.
2. Train the network using S-fold cross validation.
3. Repeat Steps 1 and 2 with different k-means clusterings.
4. Determine the average unbiased error for the network structure.
5. Repeat Steps 1–4 with different values of H or P. Search for a minimum of the average error. Select the optimal structure as the one with the minimum cross-validation error.
6. Train multiple networks with the optimal structure using the entire training set. Choose the one with the minimum training error.
7. For this network calculate n_h, V_h, ρ_h, s_{hi}^2 and CL_{hi} (or CL'_{hi}). E_{ik} are available from the training procedure at Step 5.
8. Augment the network as shown in Fig. 3.

4. RELIABILITY MEASURES FOR CLASSIFICATION

In addition to their use as nonlinear models, neural networks are also used as trainable pattern classifiers. The primary goal is to assign a "pattern" to one or more of a finite set of prespecified classes. The pattern is a vector of features containing information useful in identifying the correct class(es) of a case. The standard formulation of a neural network classifier has the feature vector as the network input and one

output node for each possible class. The target output vector to be learned has a value of one for the output node corresponding to the correct class of the example and zeros for the other output nodes. The network is trained in the usual manner of minimizing the sum of squared residuals. Using this formulation, if the network has enough representational capacity and the data are adequately dense, the network output will represent the local relative probability densities of the classes (Kramer and Leonard, 1990). However, the criterion for predicting performance and selecting the optimal structure is not the difference between the target output and the actual output as before, but the fraction of cases the network misclassifies. The network classification is taken as the class represented by the output node with the highest value.

The accuracy of the network outputs in any given region therefore can depend heavily on the number of training examples available. In regions of the feature space with little training data, the uncertainty or variance in the network outputs increases. In regions devoid of training examples, the network output is arbitrary (the relative probability density is undefined). Therefore, a test of the local amount of training data in the vicinity of the test case is needed to verify that the classification is reliable. The extrapolation measures discussed above, namely $\rho(\mathbf{x})$ and the maximum activation of the RBF units, can indicate when the neural classifier output is unreliable.

While the extrapolation measures are useful, the corresponding measurements of the uncertainty of the outputs of the network are not. This is because in functional estimation problems the training targets represent the desired network outputs and the residual errors represent the lack of fit. In the classification problem the training targets do not represent the desired network outputs. The desired network outputs are the relative probabilities while the training targets are an encoding of a symbolic variable, the class label. By minimizing the residual error during training and selecting the network with the minimum predicted classification error, one produces a classifier which "learns" the relative probabilities. If one knew the correct relative probability of membership of each class for each training example, then the network could be trained as a functional approximation instead and the confidence limits would reflect the local variance of the probabilities of membership.

Thus, when the method presented here is used on classification problems the network does not include confidence limits on the outputs. The network has output nodes representing the M possible classes, max-act and $\rho(\mathbf{x})$.

5. EXAMPLES

The examples are intended to illustrate key features of the VI-net. The first two examples demonstrate the method applied to functional estimation problems. The third example is a classification problem. The first example is a simple 1-D problem in which the shapes of the functional estimation and the reliability functions can easily be visualized. The second example is intended to demonstrate the performance on a more practical example, a process modeling problem. The third example is a fault diagnosis classification problem.

5.1. One-dimensional example

The first example problem is to estimate a single-input, single-output function given by:

$$y(x) = 0.5 \sin(1.5\pi x + \pi/2) + 2.0 + \nu,$$

where x is the independent variable, y is the dependent variable and ν is Gaussian noise. To demonstrate how reliability of the estimation can vary locally, three cases were studied. For each case 50 training examples were used. The training set was produced by pulling samples on the interval $[-1, 1]$ according to one of two probability density functions for x. The corresponding target outputs were produced using one of two distributions of ν. In each case a series of networks were trained with the number of hidden nodes ranging from 2 to 30 with the overlap parameter $P = 2$. The results are reported for the optimal network as selected by the SFCV procedure. In all cases the value of S used was 50.

In the first case, the x-values were sampled from a uniform distribution on the interval $[-1, 1]$ and ν had a standard deviation of 0.03. The optimal network structure contained 8 RBF units although networks with 7, 9 and 10 units performed almost as well. The network fit, the reliability measures and training data are shown in Fig. 4a. The network fits the function well. The confidence limits shown are for *individual* values of y so that 95% of all values should lie within the confidence band. Using a threshold value of 0.5 for the maximum activation measure indicates the network is extrapolating for $x < -1.27$ and $x > 1.09$. The minimum value of ρ (scaled 0–1) at the original training points is 0.554. Using this as a threshold value, ρ indicates extrapolation for $x < -1.1$ and $x > 0.99$. Both measures give reasonable estimates of the original sample interval of $[-1, 1]$, with ρ being more precise.

Figure 4b shows the optimal network trained on the same x data as the previous case but where the standard deviation of ν was given by:

$$\sigma_\nu = 0.045 + 0.04x.$$

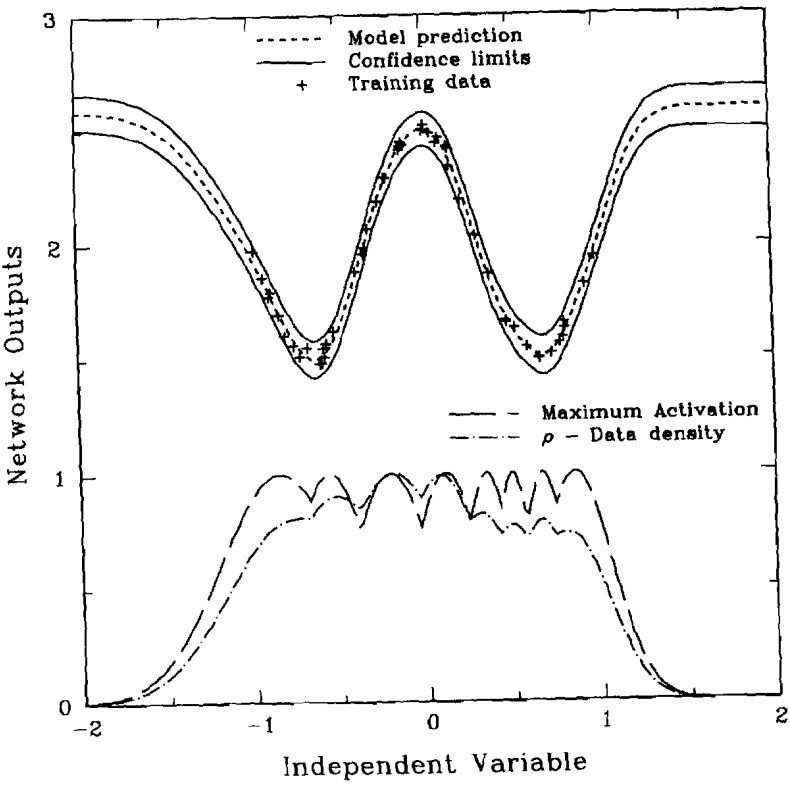

Fig. 4a. 1-D Example with uniform noise and training data distributions.

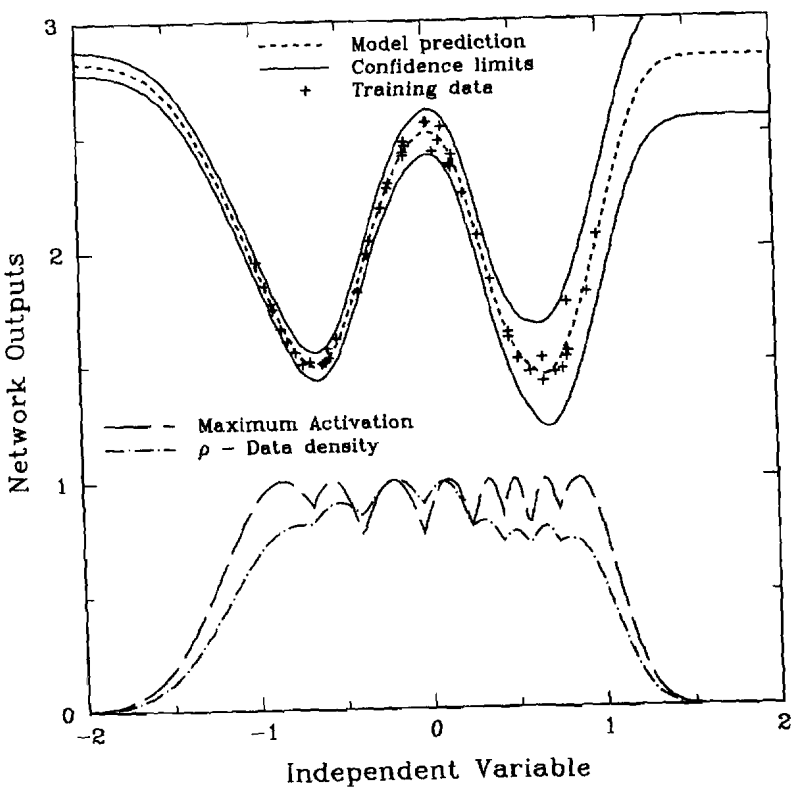

Fig. 4b. 1-D Example with triangular noise and uniform training data distributions.

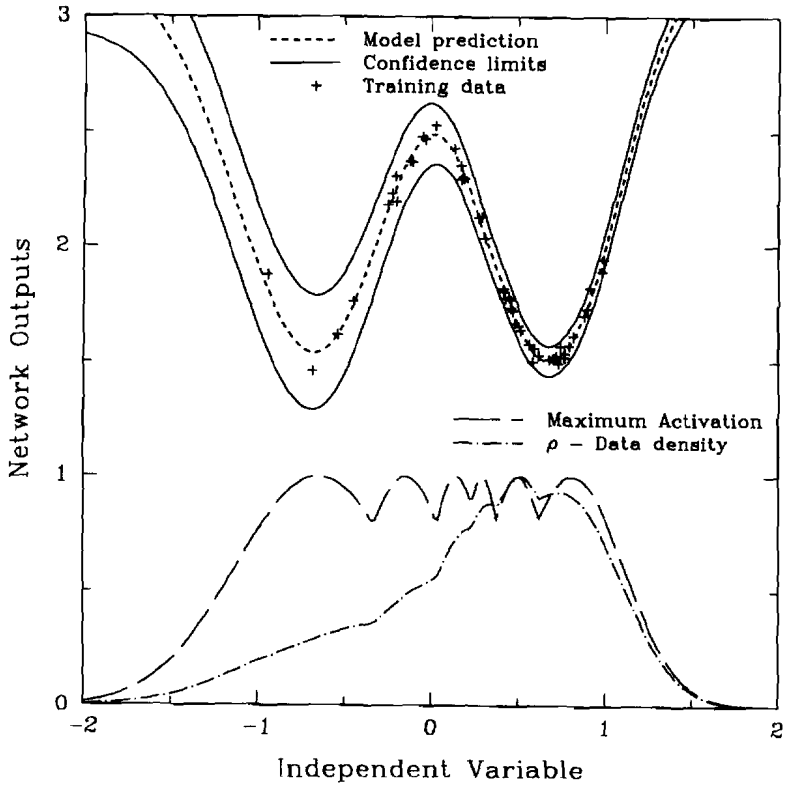

Fig. 4c. 1-D Example with uniform noise and triangular training data distributions.

The optimal number of RBF units was again 8. The confidence band in Fig. 4b shows the uncertainty in y increases as x increases, reflecting the increasing noise in y. Because the extrapolation measures are a function of only the input variables and the same x-values were used in both cases, max-act and ρ look similar in both cases.

In the third case, the noise in y was constant as in the first case, but now the x data are sampled from a nonuniform distribution with the probability density function $f(x) = 0.5 + 0.45x$ for $-1.0 \leqslant x \leqslant 1.0$ and zero otherwise. This results in having few training data near $x = -1$ and more near $x = 1$. The optimal number of hidden units for this case was six. The results are shown in Fig. 4c. Even though the noise in y is constant, the confidence band is wider near $x = -1$ than near $x = 1$, reflecting the increase in uncertainty in the estimation of y due to less training data. Notice that the skewness of the distribution of x-values is captured by ρ, but not by max-act.

5.2. *Chemical reactor model*

This problem, presented by Economou *et al.* (1986), is a multi-input, multi-output problem representing an ideal, adiabatic, continuous stirred tank reactor (CSTR) with a reversible, first-order exothermic reaction. The forward reaction, $A \rightarrow R$, is first-order in A with rate constant k_1. The reverse reaction is first-order in R with rate constant k_1. The feed is pure A and the desired product is R. The system is simulated by a set of coupled ordinary differential equations determining the reactant mass balance, the product mass balance and the energy balance:

$$dA_{out}/dt = 1/\tau(A_{in} - A_{out}) - k_1 A_{out} + k_{-1} R_{out},$$

$$dR_{out}/dt = 1/\tau(R_{in} - R_{out}) + k_1 A_{out} - k_{-1} R_{out},$$

$$dT_{out}/dt = 1/\tau(T_{in} - T_{out}) + (-\Delta H_R/\rho C_p)/(k_1 A_{out} - k_{-1} R_{out}).$$

The reaction is normally run around 400 K. The same parameter values were used as in Economou *et al.* (1986).

The goal is to learn the open-loop relationship between the controlled variable, which is the concentration of R, and the manipulated variable, which is the temperature of the feed stream T_{in}. The network also predicts the actual reactor temperature T_{out}, which is used as a check on the temperature measure-

ment. Because of the dynamics of the problem, the network must capture the state information about the reaction. This is done by giving the network a short history of the process as input, specifically, the past six measurements of T_{in} and R and the present value of T_{in}. The space-time of the reactor is 1 min and the data is sampled every 30 s so that the history input to the network is three times the characteristic time of the process which adequately captures the state of the process. (No attempt was made to optimize the input window since the objective was to demonstrate the validity measures, not necessarily to create the highest fidelity model.) The training data were generated by simulating a process run of 60 min producing 114 training examples. In order to adequately cover the normal operating domain. T_{in} was randomly varied over the range (370 K, 430 K). T_{in} was changed and new values of [R] and T_{out} were recorded every 30 s. The output targets for the network were scaled so that the model would not be overly biased toward fitting one output at the expense of the other.

About 25 RBFN architectures were trained using the procedure described in Section 3 using $S = K = 114$ (leave-one-out cross validation), so that 115 RBFNs were trained for each architecture. In these runs, the number of RBF units ranged from 2 to 90 and the overlap parameter varying from 2 to 10. The optimal network structure for this problem indicated by cross-validation was 75 RBF nodes and $P = 10$. Individual nets took about 10 s to train, and the total computational time for these runs was approx. 8 h on a VAXstation 3100 workstation (rated 0.4 MFLOPS). As in the previous example, the network performance was not overly sensitive to the number of hidden nodes. Figure 5a shows the model predictions, the confidence bands and the training data for [R]. The model fits the training data well and the confidence interval encompasses all of the training data. Figure 5b shows the model predictions, the confidence bands and the training data for T_{out}. Again the model fits the training data well and the confidence interval encompasses all of the training data. Figure 5c shows the extrapolation measures vs time on the training data. The consistently high values of the maximum activation (>0.5) indicates that all the training data are within the domain of the RBF units as expected. The average value of ρ (scaled 0–1) is approx. 0.6 with a maximum value of 0.99 and a minimum value of 0.33.

To demonstrate that the network can detect extrapolation, a testing data set was generated where the nominal target value of T_{in} was not constant but increasing by 1.5 K per minute after 15 min of normal operation. Beyond 22 min the model no longer accurately predicts [R] or T_{out} (Fig. 6). This is because the process is moving into a region for which the model was not adequately trained. This is indicated by ρ dropping to a value less than 0.33. The maximum activation is approx. 0.5 from 22 to 30 min while $\rho < 0.33$ indicates the

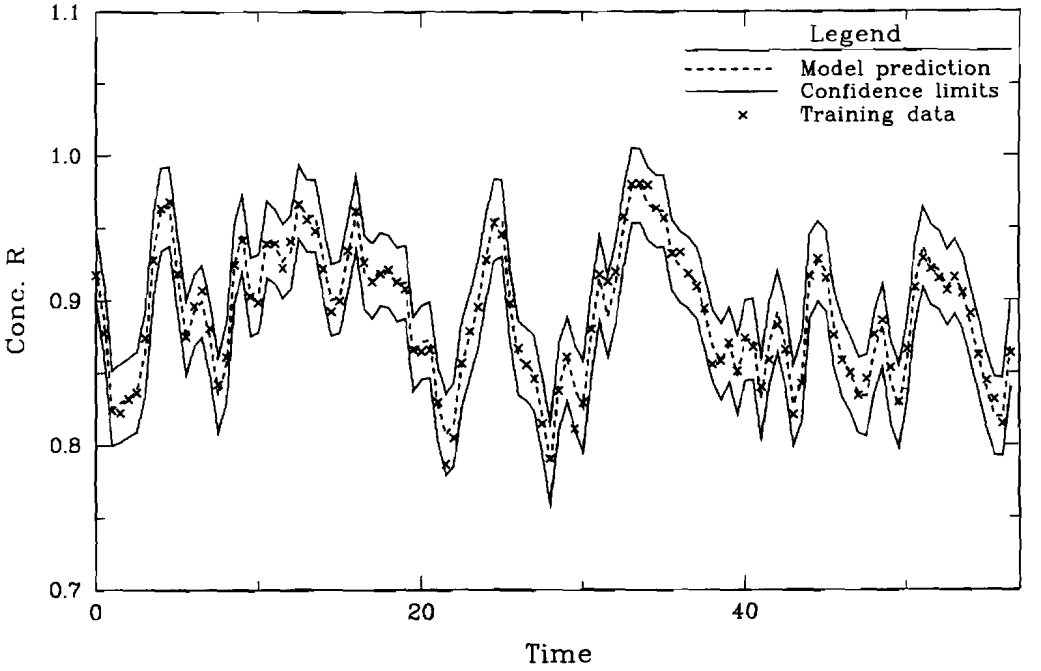

Fig. 5a. CSTR model performance on training data: concentration prediction with confidence interval on individuals.

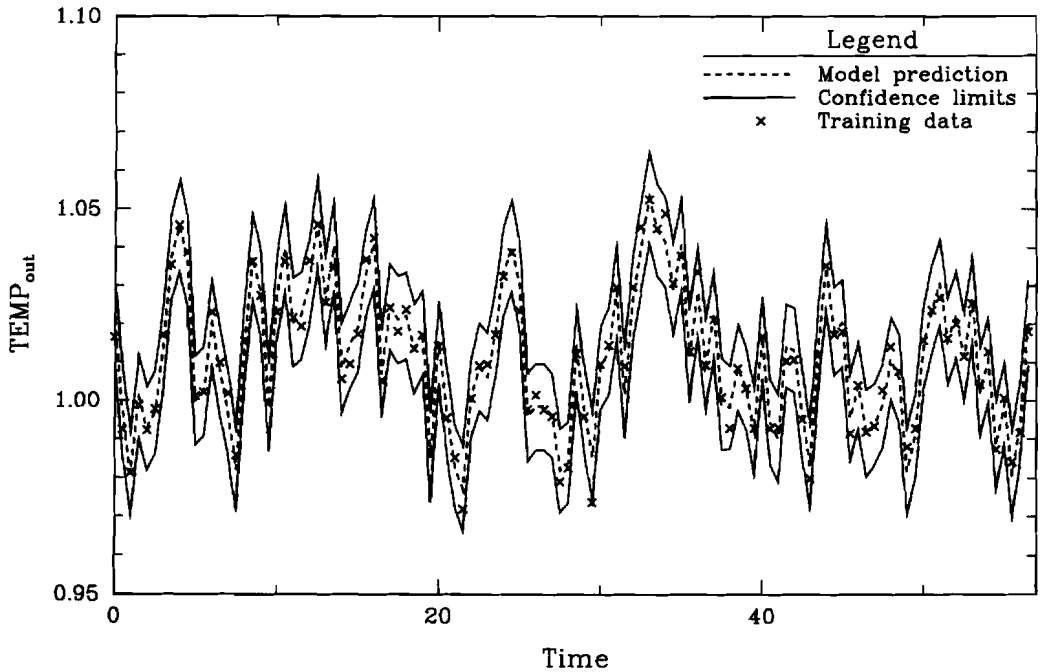

Fig. 5b. CSTR model performance on training data: temperature prediction with confidence interval on individuals.

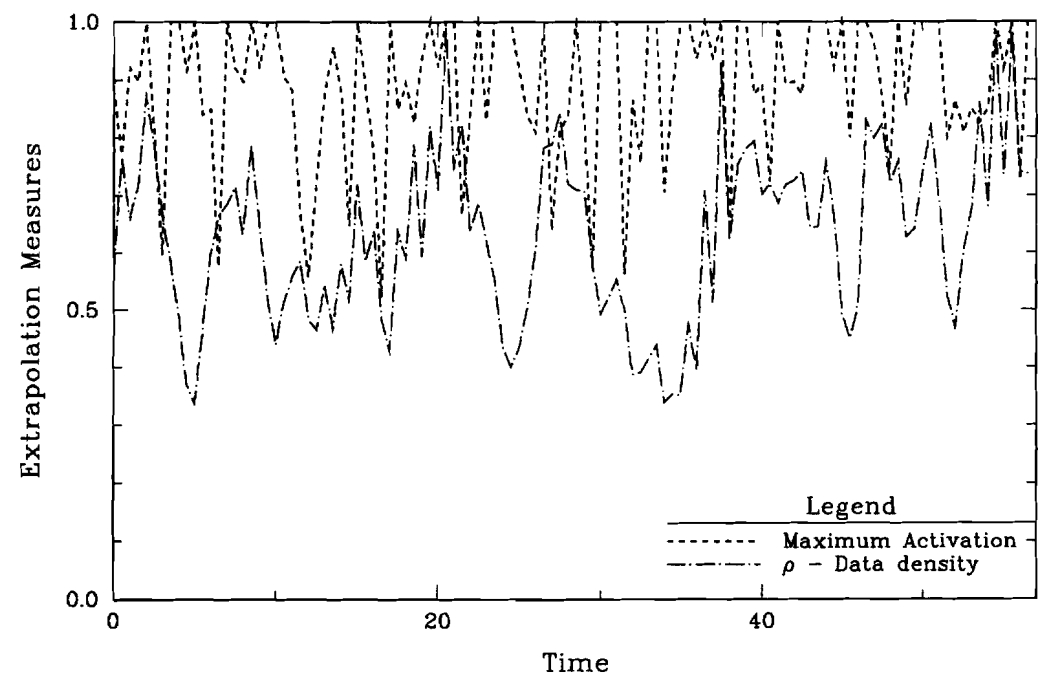

Fig. 5c Extrapolation measures for training data on CSTR model.

model is being used for extrapolation or in a region with too little training data. After 30 min the maximum activation drops below 0.5 indicating extrapolation. While both ρ and max-act indicate extrapolation, ρ is the more sensitive indicator since it reflects movement into a region of sparse data which occurs before the process leaves the region covered by radial units.

A failure to detect extrapolation in this case would be quite detrimental. When the reactor temperature

is above 434 K, the steady-state gain between T_{in} and [R] changes sign because the reverse reaction is favored at higher temperatures. An empirical model that did not include training data in the high-temperature regime, if used for model-based control, could result in the controller making corrections in the wrong direction. Using the extrapolation index, measures can be taken so the model is used only in regimes where it is known to be valid.

5.3. *Fault diagnosis classification example*

This example, a simplified version of many static diagnosis problems, has been presented previously by Kramer and Leonard (1990). The measured states of

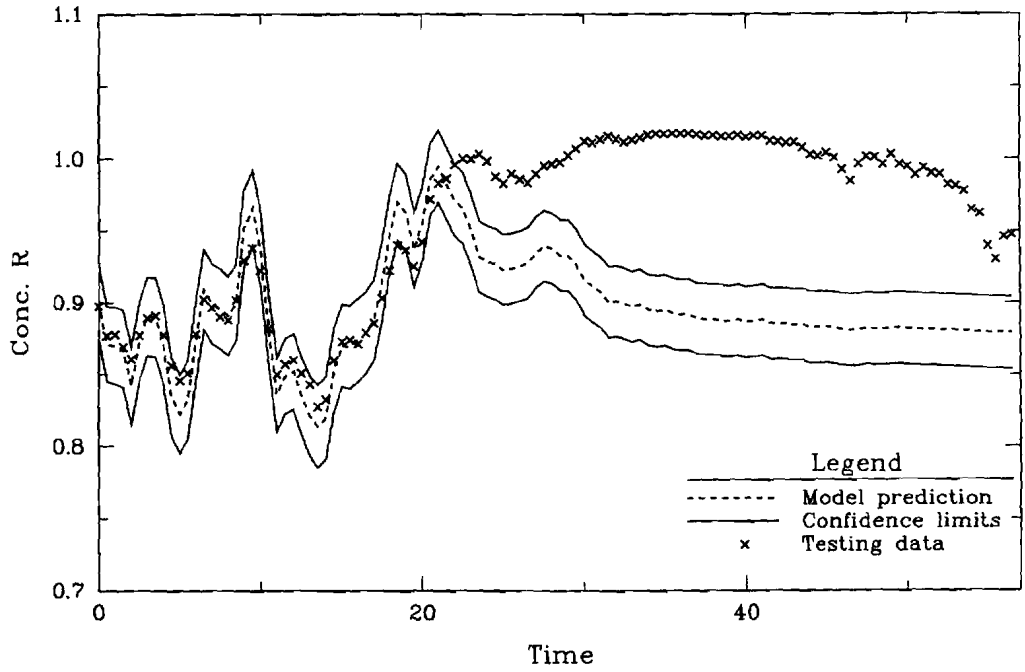

Fig. 6a. CSTR model performance on ramp testing data: concentration prediction.

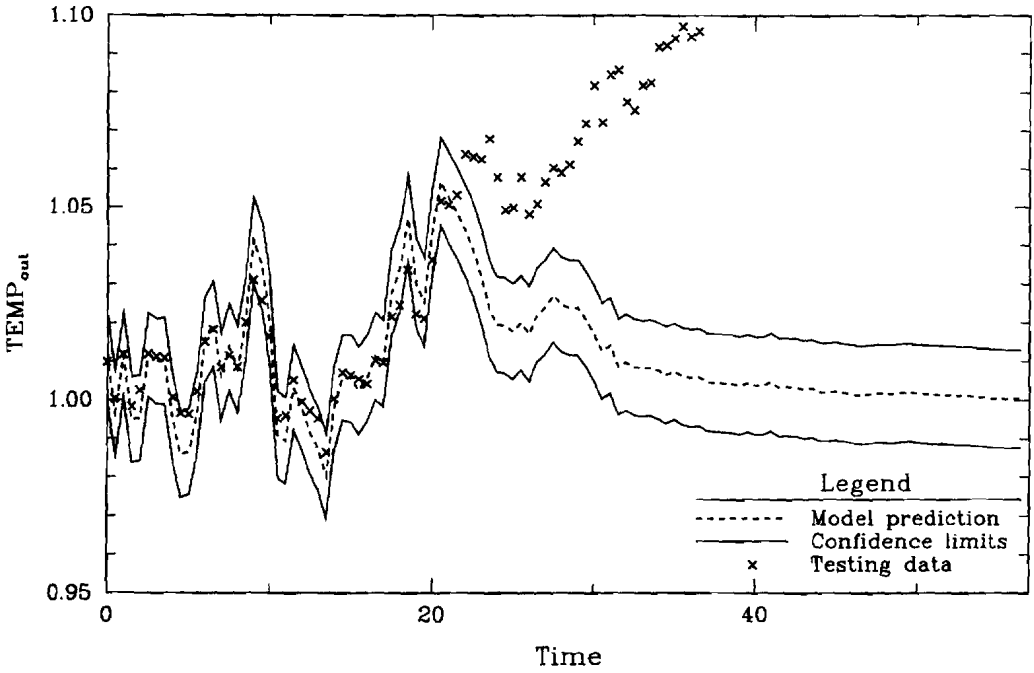

Fig. 6b. CSTR model performance on ramp testing data: temperature prediction.

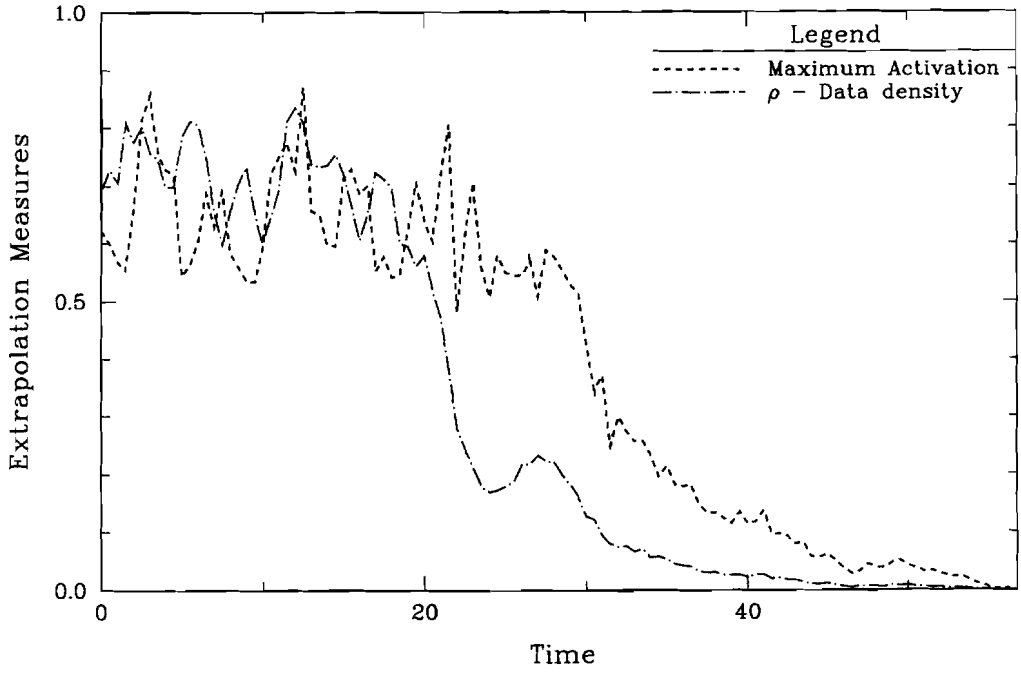

Fig. 6c Extrapolation measures for ramp testing data on CSTR model.

the process are represented by **x**, a dimension-two vector. The goal is to associate the measurements **x** with a malfunction class. Malfunctions are represented by one-at-a-time changes in a set of fault parameters **p** whose effect on the measurements x is linear:

$$\mathbf{x} = \mathbf{x}_0 + \underline{\alpha}\,\mathbf{p} + \boldsymbol{\nu}$$

Each column of $\underline{\alpha}$ represents the directional effect of one of the fault parameters on the measurement vector. The process has a nominal steady state at $\mathbf{x}_0 = (0, 0)$. The fault classes are defined as:

normal (C_0): $|p_1| < 0.05, |p_2| < 0.05$,
fault 1 (C_1): $|p_1| < 0.05$, $\boldsymbol{\alpha}_1 = [1, 1]^T$,
fault 2 (C_2): $|p_2| < 0.05$, $\boldsymbol{\alpha}_2 = [1, -1]^T$,
$\sigma_\nu = 0.015$.

Training data were generated by sampling values of p_i from a normal distribution with standard deviation 0.25. If $|p_i| < 0.05$, the example was considered to be in the normal class. Thirty examples of each class were produced. Each class is represented by a separate output node. The training targets are introduced as unit vectors, with a one in the output position corresponding to the class of the training example, and zeros on the other outputs. There are two inputs, corresponding to x_1 to x_2. In application, the output node with the maximum activation is taken as the classification. Leonard and Kramer (1991) detail the application of RBFs to this example. The emphasis here is on detection of novel cases through the extrapolation measure ρ.

The optimal network structure was found to be 15 hidden nodes but the minimum of the SFCV curve was broad and flat with networks containing 9–30 hidden nodes achieving approximately the same minimal error rate. Weiss and Kulikowski (1990) recommended choosing a structure on the low side of the minimum to prevent overfitting the data. However, this preference does not take into account how well the reliability measures are estimated. In general, we have observed that the reliability measures tend to improve as the number of hidden units increases, in the range of near-optimal structures. Figure 7a shows the decision surfaces and boundaries of the extrapolation measures defined by max-act = 0.5 and $\rho = 0.036$ (the minimum over the training data) using 15 hidden nodes, while Fig. 7b shows the results using 30 hidden units (in this case the minimum training density contour $\rho = 0.013$ is shown). For the network with 15 hidden units the extrapolation boundaries are a coarse estimate of the underlying distribution of the training points. The extrapolation boundaries for the network with 30 hidden units does a better job of capturing the shape of the parent distribution of training points. The sensitivity of the

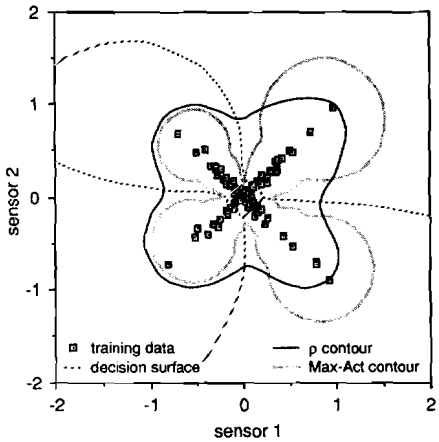

Fig. 7a. Extrapolation measure contours (15 hidden nodes).

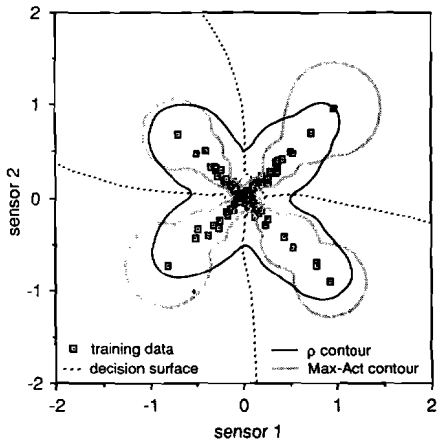

Fig. 7b. Extrapolation measure contours (30 hidden nodes).

extrapolation boundaries to the number of hidden nodes is the same type of effect seen with varying window width in the Parzen windows approach to nonparametric density estimation.

6. CONCLUSIONS

Interest has focused on using neural networks to form empirical models for engineering applications because of their powerful representational capacity and ability to handle multi-input, multi-output problems. However, conventional neural networks do not flag the user when they make predictions without adequate training data, and furthermore they do not provide error estimates on their predictions. We have presented novel network, the VI-net, that indicates extrapolation and calculates confidence limits on its predictions. These measures have been demonstrated on both functional estimation and classification problems. On several examples, the proposed measures successfully predict the quality of the model fit within the range of adequate training data and diagnose when the network encounters novel inputs for which adequate training data was not available.

The VI-net is implemented by adding connections and nodes to an RBFN that is trained in the manner described by Moody and Darken (1989). The additional weights of the VI-net are based on statistics produced during the training and cross-validation procedures of the basic RBFN model. The VI-net is produced with minimal extra work because no additional training is required, and all parameters in the underlying RBFN remain the same.

While much previous interest has focused on BPNs, these networks have no inherent ability to indicate when they are functioning outside the domain over which they were trained. This is because the contours of constant hidden node activation are hyperplanes and therefore infinite in length; their activation does not represent proximity to training data. In fact, if there is a significant hidden node activation at any point, then there is an equal activation arbitrarily far away. RBFNs, on the other hand, have an intrinsic ability to indicate when they are extrapolating since the activation of the RBF units is directly related to the proximity of the test point to the training data. BPNs have no similar mechanism to define local neighborhoods or to estimate the amount of data within an arbitrarily defined neighborhood. As a result, it is difficult or perhaps impossible to implement extrapolation measures within BPNs. If one attempted to train a BPN to recognize extrapolation, a comprehensive set of examples representing extrapolation would be needed. Forming such a training set is tantamount to solving the extrapolation/interpolation problem in advance, not to mention the practical difficulty of introducing a potentially huge set of new training examples to represent the parts of the independent variable space not covered by the original training examples. It is, however, conceivable to add auxiliary outputs to a BPN to produce the output confidence intervals. These nodes would be trained using the cross-validation errors from the original network training as targets. However, this approach is not as satisfactory as the VI-net because of the need for additional training.

NOMENCLATURE

A = Chemical species (Example 2)
a_h = Activation of radial basis unit h
CL_{hi} = 95% confidence limit (1/2 interval) of the mean value of model prediction i for radial basis unit h
$CL_i(\mathbf{x})$ = 95% confidence limit (1/2 interval) of the mean value of model prediction i at point $\mathbf{x}$

CL'_{hi} = 95% confidence limit (1/2 interval) of the individual values of model prediction i for radial basis unit h
$CL'_i(\mathbf{x})$ = 95% confidence limit of the individual values of model prediction i at point $\mathbf{x}$
H = Number of radial basis nodes in the hidden layer
K = Total number of training examples
M = Nominal number of network outputs, model predictions
N = Number of input nodes to the network
n_h = Number of training points contributing to radial basis unit h
P = Overlap parameter for determining σ_h via nearest neighbor heuristic
R = Chemical species (Example 2)
[R] = Concentration of species R (Example 2)
s^2_{hi} = Estimate of the variance of the fit of radial basis unit h for model prediction i
T_{in} = Temperature of the feed (Example 2)
T_{out} = Temperature of the effluent (Example 2)
$\mathbf{T}_k$ = Target vector for kth training example
t_{95} = Student's t-statistic for 95% confidence
w_{hi} = Connection weight for the second layer connection from hidden unit h to output node i
$\mathbf{x}$ = Arbitrary point in input space to network
$\mathbf{x}_k$ = Input vector of the kth training example
$\mathbf{x}_h$ = Location of the center of radial basis function unit h in the input space
$\mathbf{y}$ = Output of network
$\mathbf{y}_k$ = Output of network for kth training example

Greek symbols

ρ = Training data density function
σ_h = Width of radial basis unit h
σ_v = Standard deviation of Gaussian noise in dependent variables of examples

REFERENCES

Admoaitis R. A., R. M. Farber, J. L. Hudson, I. G. Kevrekidis, M. Kube and A. S. Lapedes, Application of neural nets to system identification and bifurcation analysis of real world experimental data. Los Alamos National Laboratory, Technical Report LA-UR-90-515 (1990).

Bhat N., P. Minderman, T. McAvoy and N. S. Wang, Modeling chemical process systems via neural computation. *IEEE Control Syst. Mag.* **8,** 24–30 (1990).

Chand D. R. and S. S. Kapur, An algorithm for convex polytopes. *J. ACM* **17,** 78–86 (1978).

Cybenko G., Approximation by superpositions of a sigmoidal function. *Math. Control. Sig. Syst.* **2,** 303–314 (1989).

Donat J. S., N. Bhat and T. J. McAvoy, Optimizing neural net based predictive control. *Proc. Am. Control Conf.*, pp. 2466–2471 (1990).

Duda R. O. and P. E. Hart, *Pattern Analysis and Scene Classification.* Wiley-Interscience, New York (1973).

Economou C. G., M. Morari and B. O. Palsson, Internal model control. 5. Extension to nonlinear systems. *Ind. Engng. Chem. Process. Des. Dev.* **25,** 403–411 (1986).

Haesloop D. and B. R. Holt, A neural network structure for systems identification. *Proc. Am. Control Conf.*, pp. 2460–2465 (1990).

Hernandez E. and Y. Arkun, Neural network modeling and an extended DMC algorithm to control nonlinear systems. *Proc. Am. Control Conf.*, pp. 2454–2459 (1990).

Holcomb T. and M. Morari, Local training for radial basis function networks: towards solving the hidden unit problem. *Proc. Am. Control Conf.*, pp. 2331–2336 (1991).

Hoskins J. C. and D. M. Himmelblau, Artificial neural network models of knowledge representation in chemical engineering. *Computers chem. Engng* **12,** 881–890 (1988).

Jokinen P. A., Continuously learning nonlinear networks with dynamic capacity allocation. Ph. D. Thesis, Tampere Univ. of Technology, Finland (1991).

Kramer M. A. and J. A. Leonard, Diagnosis using backpropagation neural networks: analysis and criticism. *Computers chem. Engng* **14,** 1323–1338 (1990).

Lapedes A. and R. Farber, Nonlinear signal processing using neural networks: prediction and system modeling. Los Alamos National Laboratory Technical Report LA-UR-87-2662 (1987).

Leonard J. A. and M. A. Kramer, Radial basis function networks for classifying process faults. *IEEE Control Syst. Mag.* **11,** 31–38 (1991).

Levin E., Modeling time varying systems using a hidden control neural network architecture. *Proc. Sixth Yale Workshop on Adaptive and Learning Systems*, pp. 127–132 (1990).

Ljung L., *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, NJ (1987).

MacQueen J., Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. Math. Stat. and Prob.* (L. M. Cain and J. Neymen, Eds), pp. 281–297. University of California, Berkeley Press (1967).

Medgassy, P., *Decomposition of Superposition of Distribution Functions.* Hungarian Academy of Sciences. Budapest (1961).

Moody J. and C. J. Darken, Fast learning in networks of locally tuned processing units. *Neur. Comput.* **1,** 281–294 (1989).

Narenda K. S. and K. Parthasarathy, Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks* **1,** 4–27 (1990).

Negoita C. V., *Expert Systems and Fuzzy Systems.* Benjamin Cummings, Menlo Park, CA (1985).

Parthasarathy K. and Narendra K. S. Stable adaptive control of a class of discrete-time nonlinear systems using radial basis neural networks. Report 91-03 Yale University Department of Electrical Engineering (1991).

Park J. and I. W. Sandberg, Universal approximation using radial-basis function networks. *Neur. Comput.* **3,** 246–257 (1991).

Parzen E., On estimation of a probability density function and mode. *Ann. Math. Statist.* **33,** 1065–1076 (1962).

Psichogios D. C. and L. H. Ungar, Direct and indirect model based control using artificial neural networks. *Ind. Engng Chem. Res.* (1991).

Poggio T. and F. Girosi, Regularization algorithms for learning that are equivalent to multilayer networks. *Science* **247,** 978–982 (1990).

Renals S. and R. Rohwer, Phoneme classification experiments using radial basis functions. *Proc. Int. Joint Conf. Neural Networks*, pp. 461–467 (1989).

Sanger T. D., A tree-structured algorithm for reducing computation in networks with separable basis functions. *Neur. Comput.* **3,** 67–78 (1991).

Specht D. F., Probabilistic Neural networks. *Neur. Networks* **3,** 109–118 (1990).

Tråvén H. G. C., A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions. *IEEE Trans. Neur. Networks* **3,** 366–377 (1991).

Venkatasubramanian. V., R. Vaidyanathan and Y. Yamamoto, An analysis of the learning, recall, and generalization characteristics of neural networks for process fault diagnosis. *Computers chem. Engng* **14,** 699–712 (1990).

Watanabe K., I. Matsuura, M. Abe, M. Kubota and D. M. Himmelblau, Incipient fault diagnosis of chemical processes via artificial neural networks, *AIChE Jl* **35,** 1803–1812 (1989).

Weiss S. M. and C. A. Kulikowski, *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA (1991).

Willis M. J., C. DiMassimo, G. A. Montague, M. T. Tham and A. J. Morris, Artificial neural networks in process engineering. *IEE Proc.-D.* **138,** 256–266 (1991).

Yao S. C. and E. Zafiriou, Control system sensor failure detection via networks of localized receptive fields. *Proc. Am. Control. Conf.*, San Diego (1990).

Ydstie B. E., Forecasting and control using adaptive connectionist networks. *Computers chem. Engng* **14,** 683–599 (1990).