Reducing Data Dimensionality through Optimizing Neural Network Inputs

Shufeng Tan and Michael L. Mavrovouniotis
Dept. of Chemical Engineering, Northwestern University, Evanston, IL 60208

A neural network method for reducing data dimensionality based on the concept of input training, in which each input pattern is not fixed but adjusted along with internal network parameters to reproduce its corresponding output pattern, is presented. With input adjustment, a properly configured network can be trained to reproduce a given data set with minimum distortion; the trained network inputs provide reduced data.

A three-layer network with input training can perform all functions of a five-layer autoassociative network, essentially capturing nonlinear correlations among data. In addition, simultaneous training of a network and its inputs is shown to be significantly more efficient in reducing data dimensionality than training an autoassociative network. The concept of input training is closely related to principal component analysis (PCA) and the principal curve method, which is a nonlinear extension of PCA.

Introduction

Process data are the foundation of process monitoring, evaluation, and control. Advancements in automation allow the collection of large volumes of process data. A chemical process may be equipped with hundreds or even thousands of sensors with sampling intervals of seconds or minutes. As an important step toward process understanding, engineers need to uncover the significant patterns hidden in process data. Dimensionality reduction is a way of summarizing information carried by a large number of observed variables with a few latent variables. Through dimensionality reduction, we obtain not only a reduced data set but also a model that relates all observed variables to a few latent variables; such a model is valuable to many types of data screening tasks, for example, data noise reduction, missing sensor replacement, gross error detection and correction, and fault detection.

Given an $m \times n$ matrix representing m measurements made on n variables, n is the observed dimensionality of the data set. Reduction of data dimensionality aims to map the original data matrix to a much smaller matrix of dimension $m \times f (f \ll n)$, which is able to reproduce the original matrix with minimum distortion through a demapping model (Figure 1). The dimensionality reduction is useful when there exist correlations among the observed variables. The reduced matrix describes latent variables extracted from the original matrix. Ideally, the f latent variables should retain all nonrandom variations in the observed variables; the mapping and demapping models should capture relationships between the observed variables and the latent variables.

Reduction of data dimensionality can be used to extract useful information from process data involving large numbers of measured variables. Principal component analysis (PCA) is the most commonly used technique, and has been shown to facilitate many types of data analysis in process engineering, including data validation and fault detection (Wise and Ricker, 1989), quality control (MacGregor, 1989), data visualization (Stephanopoulos and Guterman, 1989), and process monitoring (Piovoso et al., 1991; Raich and Cinar, 1994).

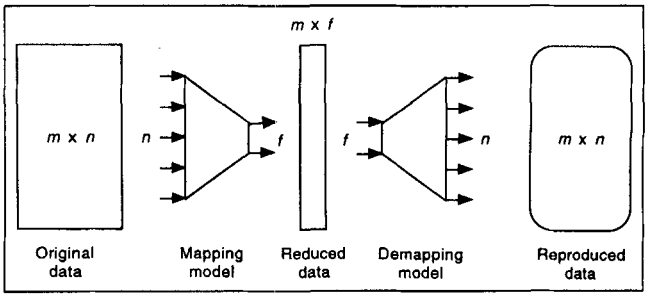

Figure 1. Reduction of data dimensionality.

Correspondence concerning this article should be addressed to M. L. Mavrovouniotis.

Singular value decomposition (SVD) provides a computationally efficient method for PCA. Any $m \times n$ matrix A of rank r can be decomposed into the following form (Strang, 1988):

$$A = u_1 s_1 v_1^T + u_2 s_2 v_2^T + \cdots + u_r s_r v_r^T \quad (s_1 \geq s_2 \geq \cdots \geq s_r > 0)$$

where s_i $(i = 1,2,\ldots,r)$ are positive scalars in descending order, u_i $(i = 1,2,\ldots,r)$ are $m \times 1$ orthonormal vectors and v_i $(i = 1,2,\ldots,r)$ are $n \times 1$ orthonormal vectors. The first f terms of the preceding decomposition provide the best approximation to A with f principal components.

PCA is a linear technique in the sense that it uses linear functions to model relationships between observed variables and latent variables. Factor analysis (FA) is another linear technique for dimensionality reduction. Both PCA and FA try to represent some aspect of the covariance (or correlation) matrix of the observed data as well as possible. PCA concentrates on the variances of individual variables, whereas in FA the interest is in the covariances of different variables (Jolliffe, 1986).

The linear techniques reduce data dimensionality by exploiting linear correlations between observed variables. If there exist nonlinear correlations between observed variables, as usually occurs in chemical processes, a nonlinear method will describe the data with greater accuracy and/or by fewer latent variables than a linear method. The necessity of nonlinear methods has long been recognized. Early research was focused on the use of polynomials and resulted in two nonlinear methods: polynomial FA (McDonald, 1967) and polynomial PCA (Carroll, 1969). A major difficulty with these methods is that polynomials represent only a very limited class of nonlinear functions.

The first principal component from PCA can be viewed as a straight line that summarizes data by minimizing the orthogonal deviation of the data from the line. Hastie and Stuetzle (1989) extended the concept of principal components to nonlinear situations, where a data set is summarized by a principal curve that minimizes the orthogonal deviation of the data from the curve. A principal curve is a set of functions driven by one parameter. The parameter is a latent variable. When two latent variables are needed, the data are summarized by a principal surface, which is a set of functions driven by two independent parameters. Principal curves and principal surfaces are determined through iterative algorithms using scatterplot smoothers.

Dimensionality reduction can also be performed by autoassociative neural networks, which are feedforward neural nets trained to perform the identity mapping between network input and output. Dimensionality reduction is achieved through a bottleneck, that is, a hidden layer with a small number of nodes. Most previous work focused on single-hidden-layer networks (Ackley et al., 1985; Cottrel et al., 1987; Abbas and Fahmy, 1993). Kramer (1991) pointed out that the single-hidden-layer architecture was unable to model nonlinear relationships between observed variables and latent variables, and consequently offered no significant improvement over PCA. He then established a three-hidden-layer architecture for autoassociative networks to capture nonlinear correlations. The three-hidden-layer autoassociative networks can be used to perform various data screening tasks, such as data noise filtering, missing measurement replacement, and gross error detection and correction (Kramer, 1992).

An autoassociative network is composed of a mapping subnet and a demapping subnet, each of which is a single-hidden-layer network by itself. Dong and McAvoy (1993) proposed to train the two subnets separately. The method they advance involves three steps: (1) find principal curves by successively applying the algorithm of Hastie and Stuetzle (1989) to observed data and residuals; (2) train a network that maps the original data to principal curves; (3) train another network that maps principal curves to the original data.

This article presents a new method for reducing data dimensionality using neural networks as nonlinear models for observed variables and latent variables. With this method, only one single-hidden-layer network is needed for dimensionality reduction of a given data set. We start with a simple example to illustrate a difficulty with training autoassociative networks. Then we introduce the concept of input training for neural networks to overcome the difficulty and derive the steepest descent direction for training network inputs. A number of data sets from mathematical models and real measurements are used to compare this network approach with other methods.

Problem with Training Autoassociative Networks

Autoassociative networks are typically trained through backpropagation. Kramer (1991) pointed out that three hidden layers are needed for an autoassociative network to provide general nonlinear mapping and demapping functions. In general, the performance of backpropagation deteriorates as the number of hidden layers gets larger (Hertz et al., 1991). The following example illustrates the difficulty with training an autoassociative network.

Example 1: circle data

Kramer (1991) examined a three-hidden-layer autoassociative network with a set of data points from a circle in two dimensions:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = 0.8 \begin{pmatrix} \sin(\lambda) \\ \cos(\lambda) \end{pmatrix}$$

where λ is uniformly distributed on $[0, 2\pi)$.

Using backpropagation, we train an autoassociative network to reproduce 20 samples of $[z_1 \ z_2]$. The network structure is sketched in Figure 2 and denoted with the numbers of nodes in different layers from input to output. All nodes use a sigmoidal transformation function, tanh(.). In the plotting of the training result (Figure 3), the data reproduced by the network are on the same circle as the original data, but there are several points that significantly deviate from the original data around the circle. Two data points near $(-0.7, -0.4)$ are projected to the same position by the autoassociative network. This type of discrepancy is persistent in spite of changes in initial weights and biases. A similar phenomenon is also observed in Kramer's results, which contain 100 data points. Therefore, it is necessary to investigate the reason behind the observed pattern of discrepancy.

We used a set of uniformly distributed data points to test the trained autoassociative network. When an input pattern

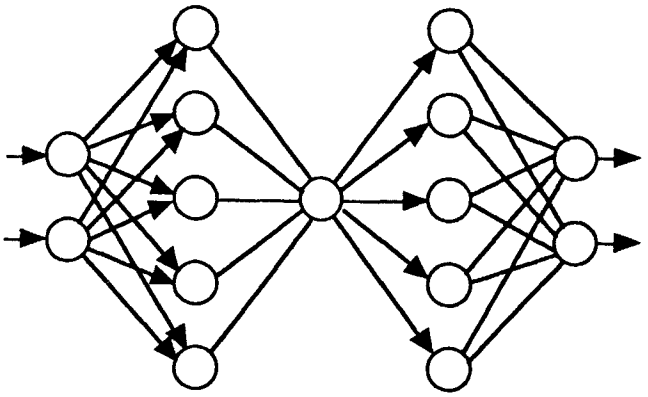

Figure 2. A 2-5-1-5-2 autoassociative network.

is fed into the trained network, it is first transformed to a single number by the mapping subnet; the number is then used by the demapping subnet to produce an output pattern. In testing results of Figure 4, we also draw lines to associate testing data points with their corresponding network outputs. The network did a fairly good job in producing points on the circle even for input points that are far away from the curve, indicating that the demapping subnet was trained to produce the correct type of output. It is the mapping subnet that fails to generate correct numbers at the bottleneck. Thus the mapping subnet does not produce a good λ, and the demapping subnet translates the bad λ into a bad result, which nevertheless lies on the circle.

The poor performance of backpropagation in training the mapping subnet is attributable to the large number of layers. In the process of backpropagation learning, modifications of weights are based on errors propagated backward from the output layer. After several layers of error propagation, the searching direction for the weights in the mapping subnet may deviate from the direction that minimizes the output error

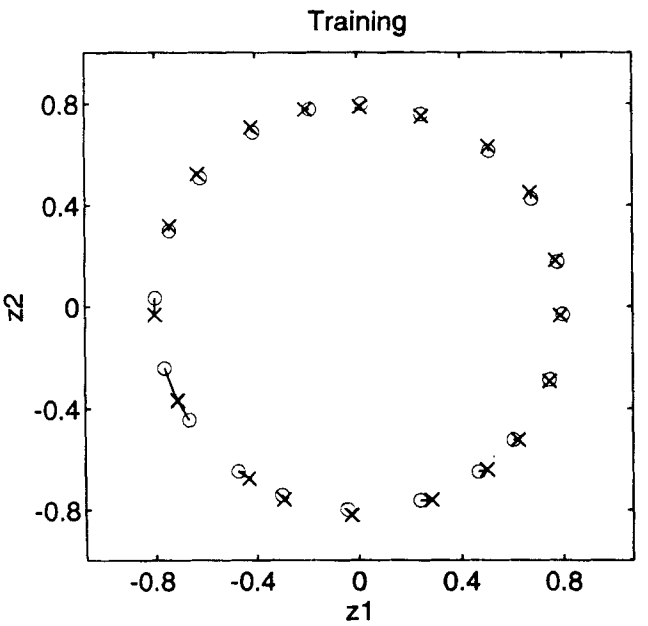

Figure 3. Training results of autoassociative network for example 1.

○ = Training data; × = output of 2-5-1-5-2 autoassociative network.

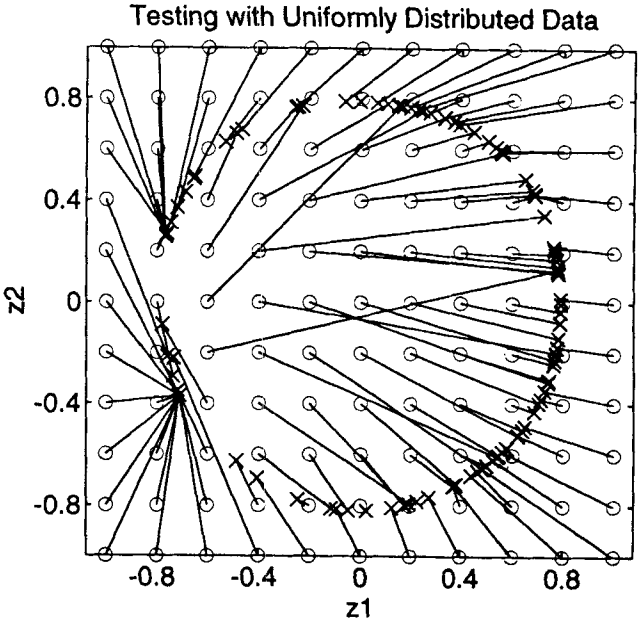

Figure 4. Testing results of autoassociative network for example 1.

○ = Testing data; × = output of 2-5-1-5-2 autoassociative network.

function. This effect becomes even more pronounced for an autoassociative network due to its bottleneck layer.

Concept of Input Training

Instead of training a whole three-hidden-layer autoassociative network, we propose to train its demapping subnet only. Training such a subnet is meaningful and can be done by extending the backpropagation algorithm, because the error function is well defined. The difference between training a demapping subnet and training an ordinary feedforward network is that the inputs to the subnet are not given. We shall adjust not only the internal network parameters but input values to reproduce the given data as accurately as possible. When network inputs are adjusted, each output sample should be uniquely associated with one input vector. Figure 5 shows a 2-4-5 network with input adjustment used for reducing the dimensionality of a data set from five to two. Each input vector $(x_{p1}\ x_{p2})^T$ is adjusted to minimize only the error of its *corresponding output* vector $(z_{p1}\ z_{p2} \ldots z_{p5})^T$ while internal network parameters are trained using *all* output samples.

After the demapping subnet and its inputs are properly trained, we obtain a reduced matrix and a demapping model in the form of a neural network. Thus all requirements for data dimensionality reduction can be fulfilled through training a single-hidden-layer network and its input simultaneously. By introducing the concept of *input training* (IT), we have actually acquired a new alternative to the autoassociative network architecture for reducing data dimensionality. To avoid confusion, we will no longer consider the type of network as a subnet of an autoassociative network. We shall refer to it as *IT-net*. Two characteristics are basic for an IT-net: the input layer has fewer nodes than any other layer, and inputs are adjusted according to corresponding outputs.

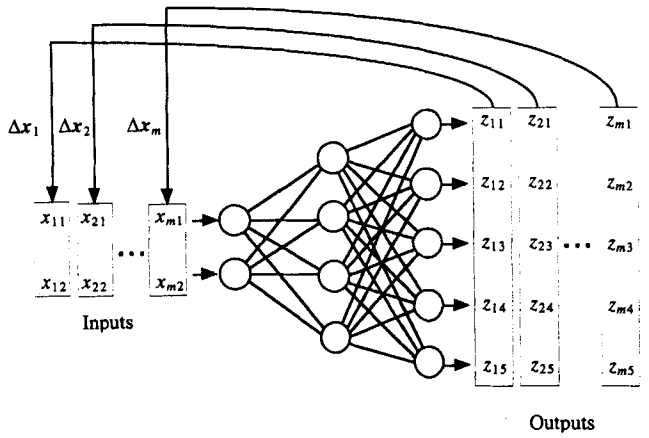

Figure 5. Concept of input training.

Note that the term *input* in the context of input training slightly differs from what is used for traditional neural networks, where inputs are always given. It is not unusual, however, to adjust inputs to a model while its parameters are being modified to minimize the output error. Examples of this model-fitting strategy include the polynomial PCA and FA reviewed in the introduction. Input training is an application of the same strategy in neural networks.

To better understand input training, we study the simplest IT-net with linear nodes only and no hidden layer. The objective of training such an IT-net is to minimize the sum of square errors:

$$\min E = \|\boldsymbol{X}\boldsymbol{W}^T - \boldsymbol{T}\|^2 \tag{1}$$

where $\boldsymbol{T}$ is the original data matrix ($m \times n$), $\boldsymbol{X}$ reduced matrix ($m \times f$), and $\boldsymbol{W}$ weight matrix ($n \times f$) of the IT-net. Note that the notation used here differs from that in the literature on autoassociative networks, where $\boldsymbol{T}$ is reduced data. To solve the minimization problem, we obtain partial derivatives of $\boldsymbol{E}$ with respect to all variables, including network inputs:

$$\frac{\partial E}{\partial \boldsymbol{W}} = \begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \cdots & \frac{\partial E}{\partial w_{1f}} \\ \cdots & & \cdots \\ \frac{\partial E}{\partial w_{n1}} & \cdots & \frac{\partial E}{\partial w_{nf}} \end{bmatrix} = 2(\boldsymbol{X}\boldsymbol{W}^T - \boldsymbol{T})^T\boldsymbol{X} \tag{2}$$

$$\frac{\partial E}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \cdots & \frac{\partial E}{\partial x_{1f}} \\ \cdots & & \cdots \\ \frac{\partial E}{\partial x_{m1}} & \cdots & \frac{\partial E}{\partial x_{mf}} \end{bmatrix} = 2(\boldsymbol{X}\boldsymbol{W}^T - \boldsymbol{T})\boldsymbol{W}. \tag{3}$$

To reach the minimum training error, all derivatives in Eqs. 2 and 3 are necessarily zero, which leads to

$$\boldsymbol{W} = \boldsymbol{T}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{4}$$

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{W})^{-1}. \tag{5}$$

A straightforward way of using these equations is to start with random input values and apply Eqs. 4 and 5 alternatively to calculate weights and new input values. This process is equivalent to the following iterative procedure:

$$\boldsymbol{X}^{(\text{new})} = \boldsymbol{T}\boldsymbol{T}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{T}\boldsymbol{T}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}. \tag{6}$$

This iteration is close in spirit to the block power method (Strang, 1988) for calculating multiple eigenvectors of $\boldsymbol{T}\boldsymbol{T}^T$. The difference is that the orthogonalization in each iteration of the block power method is replaced with right multiplication of an $f \times f$ matrix, $(\boldsymbol{X}^T\boldsymbol{T}\boldsymbol{T}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}$. Specially, if the IT-net has only one input node, $\boldsymbol{X}$ becomes a vector and the preceding iteration reduces to:

$$\boldsymbol{x}^{(\text{new})} = \frac{\boldsymbol{x}^T\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{T}\boldsymbol{T}^T\boldsymbol{x}}\boldsymbol{T}\boldsymbol{T}^T\boldsymbol{x}. \tag{7}$$

which is the power method for finding the largest eigenvalue of $\boldsymbol{T}\boldsymbol{T}^T$ using the Rayleigh quotient as the scaling factor in each iteration. It can be shown that this iteration converges to an eigenvector of $\boldsymbol{T}\boldsymbol{T}^T$ corresponding to the largest eigenvalue, which is exactly the first principal component of $\boldsymbol{T}$. Therefore, training an IT-net with one input node and no hidden layer is equivalent to PCA.

IT-nets are basically feedforward networks. With one hidden layer of sigmoidal nodes, a feedforward network can approximate any nonlinear function to an arbitrary accuracy given sufficient hidden nodes (Cybenko, 1989). Let $\phi_k(\lambda_1,\ldots,\lambda_f)$, $k=1,\ldots,n$, denote nonlinear mappings by an IT-net. When the output error for a given data vector, $(t_{p1},\ldots,t_{pn})^T$, is minimized at an input vector, $(x_{p1},\ldots,x_{pf})^T$, we have:

$$\left[\frac{\partial}{\partial\lambda_i}\sum_k(\phi_k - t_{pk})^2\right]_{\lambda_i = x_{pi}} = 0, \quad i=1,\ldots,f, \tag{8}$$

which can be rearranged into

$$\sum_k(z_{pk} - t_{pk})\left[\frac{\partial\phi_k}{\partial\lambda_i}\right]_{\lambda_i = x_{pi}} = 0, \quad i=1,\ldots,f \tag{9}$$

where $z_{pk} = \phi_k(x_{p1},\ldots,x_{pf})$.

If the IT-net has exactly one input node, the functions $\phi_k(\lambda)$, $k=1,\ldots,n$, represent a smooth curve in n-dimensional space. Equation 9 indicates that the vector of output errors, $\boldsymbol{z}_p - \boldsymbol{t}_p$, is orthogonal to the tangent of the curve at $\lambda = x_p$ when the sum of square errors is minimized through input training. Therefore, the result of training a single-input-node IT-net with a hidden layer of sigmoidal nodes is equivalent to a principal curve as defined by Hastie and Stuetzle (1989). Similarly, training a two-input-node IT-net will result in a principal surface, which is a vector of continuous functions driven by two parameters and minimizes the orthogonal deviation of the data from the surface (Hastie and Stuetzle, 1989).

Training IT-Nets

For IT-nets with hidden layers, a direct iterative procedure for network inputs such as Eq. 6 is not available. We extend

the backpropagation training method to network inputs. Similar to network weights, network inputs are modified using errors backpropagated from the output layer. The steepest descent direction for minimizing the output errors through adjustment of network inputs is derived in this section.

Let t_{pk} be the value of the kth observed variable in the pth training sample and z_{pk} the corresponding IT-net approximation. Then the objective function to be minimized in network training is

$$E = \sum_p \sum_k (z_{pk} - t_{pk})^2. \tag{10}$$

The steepest descent direction for optimizing network inputs x_{pi} is given by

$$\Delta x_{pi} = -\frac{\partial E}{\partial x_{pi}} = \sum_k (t_{pk} - z_{pk}) \frac{\partial z_{pk}}{\partial x_{pi}}. \tag{11}$$

Assuming that input and output nodes use the identity activation function, while hidden nodes use a sigmoidal function, the network output is given by

$$z_{pk} = \sum_j w_{kj} \sigma \left(b_j + \sum_i v_{ji} x_{pi} \right) \tag{12}$$

where $\sigma(.)$ is a sigmoidal function, b_j is the bias of the jth hidden node, and V_{ji} and W_{kj} are network weights. Hence, the steepest descent direction for training network inputs is

$$\Delta x_{pi} = \sum_j v_{ji} \delta_{pj} \tag{13}$$

where δ_{pj} is the propagated error at the hidden layer and has been defined as

$$\delta_{pj} = \sigma' \left(b_j + \sum_i v_{ji} x_{pi} \right) \sum_k w_{kj} (t_{pk} - z_{pk}). \tag{14}$$

Note that the steepest descent direction for training network weights between the input layer and the hidden layer is

$$\Delta v_{ji} = \sum_p x_{pi} \delta_{pj}. \tag{15}$$

Therefore, the extra computation required for training the inputs is negligible compared with training the rest of the network. In the preceding derivation, we have assumed that only hidden nodes use sigmoidal functions and that input and output nodes are linear. The same derivation can be carried out for networks with sigmoidal output and/or input nodes and Eq. 13 still holds but with different δ_{pj}.

Testing and Using IT-Nets

A trained IT-net can be tested through cross validation. In this sense, testing and using an IT-net involve the same computing task. We will need to describe only testing. Because network inputs are unknown for testing samples, the appropriate way to test a trained network is to adjust network inputs while freezing all internal network parameters (weights and biases). That is, testing an IT-net still requires a searching procedure that optimizes each input pattern to yield a good approximation for its corresponding output sample.

Note that optimization of inputs for testing is much less time-consuming than training a whole IT-net. First, testing can be done for each individual sample and it involves much fewer searching variables than training the whole network. In addition, since the inputs of training data are available, we can apply a nearest neighbor algorithm to obtain good initial guesses for the inputs of testing samples. Finally, replacing the small fixed learning rate with a 1-D search significantly improves the speed of optimizing network inputs.

As an alternative to the preceding testing method, after training an IT-net we might proceed to train another network that maps the observed data to the reduced data. An autoassociative network could then be constructed by combining the mapping network and the IT-net. The major motivation for doing this is that using a trained autoassociative network would then be as simple as feedforward calculation. However, the result of this method of construction of an autoassociative network is often disappointing due to the following factors: (1) Training errors are introduced twice; (2) two network training rounds are independent of each other and there is no effective mechanism to ensure the quality of the whole autoassociate network. All testing results in this article were obtained through optimizing network inputs in an IT-net.

Examples Using Artificial Data

We first try IT-nets on several sets of data produced from known nonlinear models. Examples involve small numbers of observed variables so that results are easy to visualize.

Example 1 revisited

We trained a 1-5-2 IT-net on the circle data to compare the IT-net with the autoassociative network approach. All nodes use sigmoidal functions. In our experience, training by backpropagation tends to converge more easily for networks using sigmoidal nodes everywhere. A different data set taken from the same circle is used for testing. Figure 6 shows that the IT-net reproduces the circle data more accurately than the autoassociative network. Training and testing results are in Table 1. Root-mean-square errors are defined as:

$$e = \sqrt{\frac{\sum_{p=1}^{m} \sum_{k=1}^{n} (z_{pk} - t_{pk})^2}{mn}}. \tag{16}$$

Since the inputs of an IT-net are allowed to change, one may suspect that such a network would reproduce any noisy or erroneous data. We can show this is not the case through numerical testing. The trained IT-net was tested with a set of uniformly distributed data. Figure 7 shows the output of the test. For each given data point, the IT-net finds a point that is on the circle and is nearest to the given point. This result demonstrates that the trained IT-net is able to find the best model-consistent estimates for noise-corrupted data.

Note that although the testing data include values outside

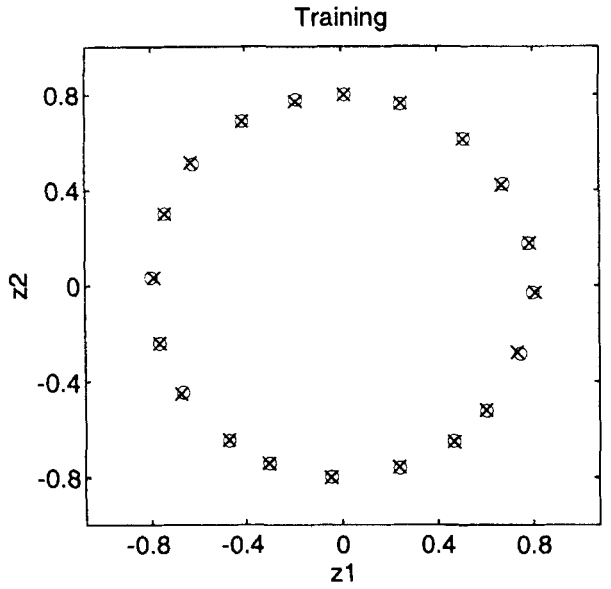

Figure 6. Training results of IT-net for example 1.

○ = training data; × = output of 1-5-2 IT-net.

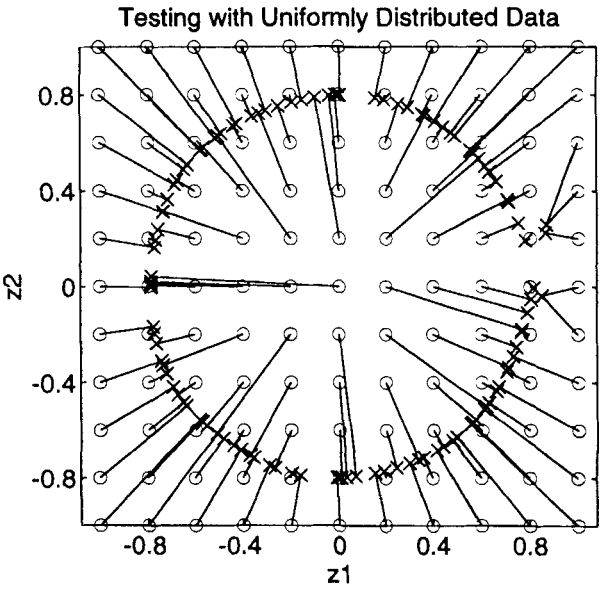

Figure 7. Testing results of IT-net for example 1.

○ = Testing data; × = output of 1-5-2 IT-net.

the training range, the IT-net does *not* extrapolate. Instead, it projects the data to produce (corrected) output values that are essentially within the training range (except for a few data points that lie slightly past the training range). This is advantageous for many applications. Five data points at $z_1 = 0.9$, $z_2 = -0.2$, 0, 0.2, 0.4, 0.6 are not projected orthogonally to the circle. This is because the mapping that the IT-net actually learned is not exactly a circle (Figure 8). The IT-net projects data points to the curve it actually learned, which deviates from the circle in small regions at the two ends.

Example 2: square data

Training data for this example are coordinates of 32 points on the borders of a square. The observed dimensionality is two and can be reduced to one. This data set is more difficult than the circle example for neutral networks because of the nonsmoothness of the square. Results of training a 1-6-2 IT-net and a 2-6-1-6-2 autoassociative network are in Figure 9. Data points for testing are taken between training data points. Table 2 shows results for training and testing the two networks. The superiority of the IT-net over the autoassociative network becomes even more significant in this example.

Example 3: circle data with Gaussian errors

This example has been used by Hastie and Stuetzle (1989) to demonstrate their principal curve algorithm. We trained an IT-net on a set of 100 data points from a circle with independent Gaussian errors in both coordinates:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \frac{2}{3}\begin{pmatrix} \sin(\lambda) \\ \cos(\lambda) \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$$

where λ is uniformly distributed on $[0, 2\pi)$ and e_1 and e_2 are independent $N(0, 0.1333)$. Another set of 100 data points were generated for cross-validation. Both data sets are severely corrupted by the random errors because the standard deviation of the noise is as much as 20% of the radius of the circle.

We use a 1-5-2 IT-net to approximate the 100 training data. The training process stopped when the error on the testing data started to increase. Training data in Figure 10 are shown as circular points and network outputs as crosses. The circle

Table 1. Root-Mean-Square Errors for Example 1

Network Type	Training Error	Testing Error
Autoassociative	0.032	0.070
Input training	0.005	0.006

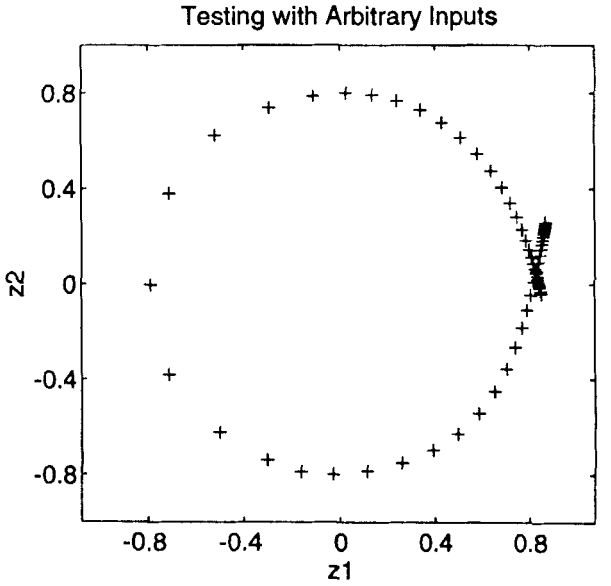

Figure 8. Actual mapping by an IT-net trained to learn the circle data.

+ = output of 1-5-2 IT-net.

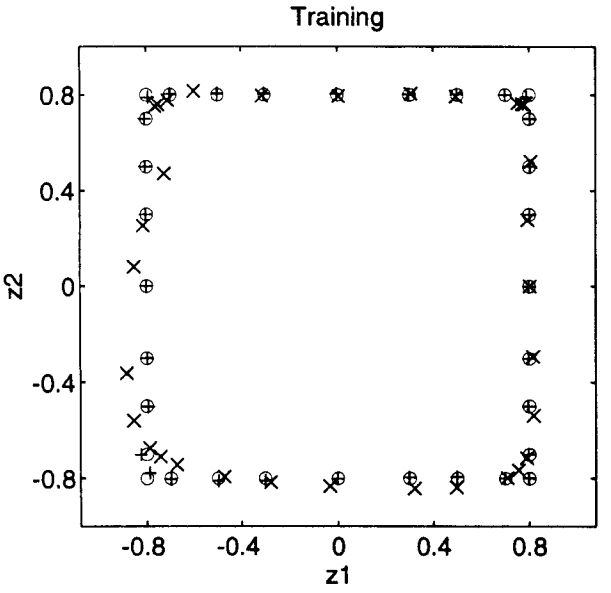

Figure 9. Training results for example 2.

○ = training data; + = output of 1-6-2 IT-net; × = output of 2-6-1-6-2 autoassociative network.

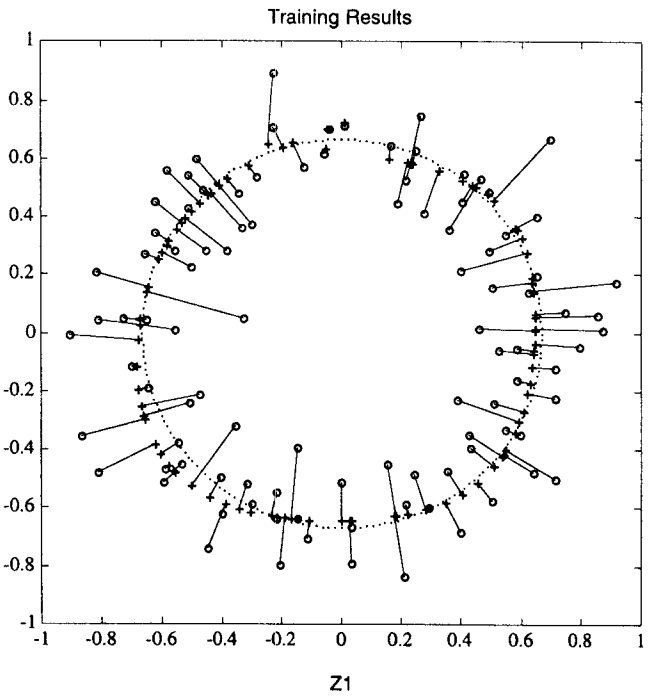

Figure 10. Training results for example 3.

○ = training data; + = output of IT-net; ... perfect circle.

used to generate the training data is drawn in a dotted line. We find that the outputs of the trained IT-net form a smooth curve that approximates the circle well. To evaluate the performance of the IT-net quantitatively, we calculated the distances from the origin to all training data points and output points. Table 3 shows the means $[E(r)]$ and the standard deviations $[\sigma(r)]$ of these distances for both training and testing sets. Note that $E(r)$ of the network outputs remains approximately equal to the radius of the circle, while $\sigma(r)$ has been reduced by 80% in the network outputs.

Application to Real Plant Measurements

Example 4: power plant data

Data in this example come from four air heaters in an electric power plant. In the flow diagram (Figure 11), two separate streams of air are heated by a stream of hot gas. Inlet and outlet temperatures of each stream are measured in each air heater to monitor the energy utilization. Figure 12 shows the sensor locations at one end of an air heater. There are 65 measurement points on all heaters and 404 data samples are available. The observed dimensionality is 65 and will be reduced to a much smaller number. Several issues need to be addressed to apply an IT-net to this set of real measurement data.

Data from real measurements usually involve a large number of variables, some of which may be linearly correlated. Since PCA is more efficient than training an IT-net for data with linear correlations, we should first separate linearly correlated variables. This is done by grouping together variables with correlation higher than a threshold value. Each group of variables can then be replaced by their first PC. The PCs of grouped variables together with the upgrouped variables are used as outputs for the IT-net. Some error is introduced when a number of variables are replaced by their first PC. This error increases as the correlation threshold decreases; therefore, the threshold should be chosen so that the error of PCA on each group of variables is smaller than that of training the IT-net. Since the IT-net error is unknown at the beginning, the threshold selection can be carried out as an iterative procedure. In this example, correlated variables are basically redundant measurements of the same variable of slightly different physical locations; thus, we can choose 0.99 correlation as the threshold for grouping correlated variables. Ten groups of variables are obtained for the given data (Table 4). Each group of variables can be reduced into a single variable through PCA. The table also shows the percentage of variation retained by the first PC of each group and the error in the reduction of each group of variables into a single PC. By replacing each group of variables with their first PC, the 31 variables in the ten groups are represented by ten PCs—a reduction of 21 variables. We will train an IT-net to further condense the remaining 44 variables (ten PCs plus 34 original variables that are not grouped).

Neither the number of hidden nodes nor the number of input nodes is known for the IT-net. The number of input nodes can be estimated from PCA results, which are readily

Table 2. Root-Mean-Square Errors for Example 2

Network Type	Training Error	Testing Error
Autoassociative	0.047	0.174
Input training	0.006	0.005

Table 3. Results for Example 3

	$E(r)$ Training	$\sigma(r)$ Training	$E(r)$ Testing	$\sigma(r)$ Testing
Training data	0.668	0.133	0.686	0.133
Output of IT-net	0.670	0.026	0.674	0.028

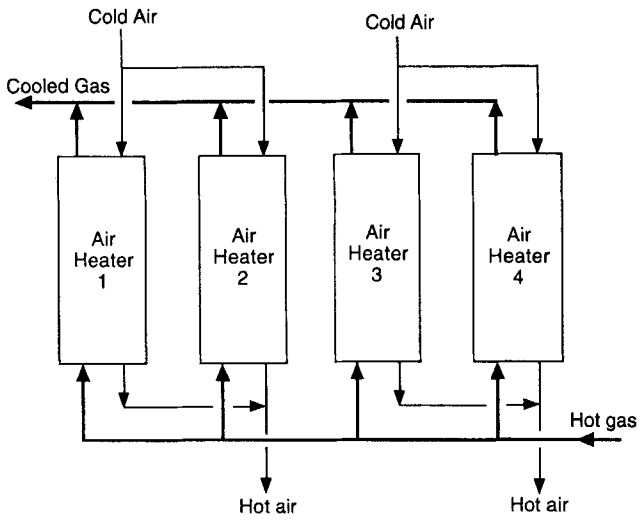

Figure 11. Flow diagram of the process in example 4.

available from SVD (Table 5). Note that the samples have been split into a training set and a testing set, and that PCA is performed on the training set only. Errors for the testing set can be computed from the loading matrices of the training set through linear regression. From the errors in Table 5, we see that two PCs retain approximately 85% of the total variation. Therefore, two input nodes are a reasonable start for the IT-net. Depending on training results, we may need to increase or reduce the input nodes.

To avoid overfitting, we should choose the number of hidden nodes so that the total number of adjustable parameters is considerably smaller than the number of training data. For an IT-net with f input nodes, h hidden nodes, and n output nodes trained on m samples of data, the number of weights is $(fh + hn)$, the number of biases is at most $(f + h + n)$, and the number of input values is fm; input values should be considered as adjustable parameters for IT-nets. Then the number of hidden nodes is limited by the following inequality:

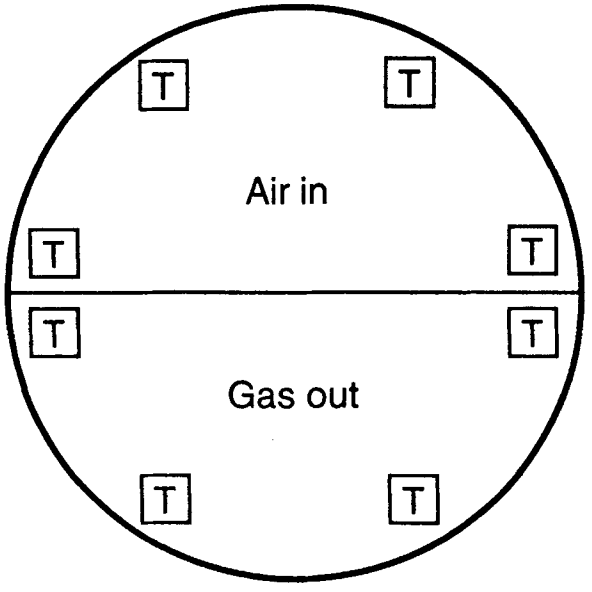

Figure 12. Temperature sensor locations on one end of an air heater.

Table 4. Groups of Linearly Correlated Variables

Group	Variables	Retained Variation	RMS Error
1	1 2 3 4	99.70%	0.011
2	5 6	99.68%	0.021
3	7 8	99.83%	0.005
4	9 10 11 12	99.63%	0.012
5	21 22 23 24	99.50%	0.008
6	33 34 35 36	99.85%	0.007
7	37 38	99.95%	0.009
8	39 40	99.83%	0.005
9	42 43 44	99.70%	0.010
10	53 54 55 56	99.46%	0.009

$$(fh + hn) + (f + h + n) + fm \ll mn. \tag{17}$$

This constant provides an upper bound for the number of hidden units. We may still need to train and compare several IT-nets with different numbers of hidden nodes. For this example, there are 8,888 values to be approximated. If the hidden layer has 66 nodes, there will be 3,552 adjustable parameters, which are reduced to 967 if the hidden layer has 11 nodes. Thus, we can in principle use as many as 66 hidden nodes and still have sufficient data to determine adjustable parameters. With 11 or fewer hidden nodes we are well within a conservative ratio of data to adjustable parameters.

Since the PCA results are available, we can use the first two columns of the score matrix as initial inputs for training the two-input-node IT-nets. Results for the IT-nets are given in Table 6. Training results are not sensitive to the number of hidden nodes when it is larger than 6. The results show that the IT-nets with two latent variables approximate the data better than PCA with three PCs.

Errors of the IT-nets can be significantly reduced by adding one more input node. Table 7 shows that the IT-nets with three latent variables reproduce the data more accurately than PCA with five PCs. The iterations necessary for training the IT-nets are given on the last rows of Tables 6 and 7. Actual training times vary for different numbers of hidden nodes. On an HP 9000/712/80i workstation, for instance, it takes 6 CPU seconds to perform one iteration for the 3-66-44 IT-net and 0.15 CPU second for the 3-6-44 IT-net. The total training time is 6.4 CPU hours for the 3-66-44 IT-net and 0.36 CPU hour for the 3-6-44 IT-net. For the trained 3-66-44 IT-net, it takes only 3 CPU seconds on average to find the input values for a given testing sample; a trained 3-6-44 IT-net takes less than 0.1 CPU second per testing sample.

To compare the 3-33-44 IT-net with PCA with three PCs, we plot the root-mean-square errors for individual output variables in Figure 13, which shows that the IT-net outperforms PCA on almost all variables. Values for some observed variables reproduced by the 3-33-44 IT-net and PCA are plotted in Figures 14 and 15 to visualize the differences in the results of the two methods.

Table 5. PCA Results for Example 4

No. of PCs	1 PC	2 PCs	3 PCs	4 PCs	5 PCs
Retained variation	66.1%	84.9%	89.8%	92.9%	95.7%
Training error	0.077	0.051	0.042	0.035	0.027
Testing error	0.077	0.052	0.042	0.035	0.028

Table 6. Results of IT-Nets with Two Input Nodes

IT-Net Structure	2-5-44	2-6-44	2-33-44	2-66-44
Training error	0.043	0.039	0.038	0.039
Testing error	0.044	0.040	0.039	0.040
Iterations	3,400	4,200	3,600	3,500

Table 7. Results of IT-Nets with Three Input Nodes

IT-Net Structure	3-6-44	3-11-44	3-33-44	3-66-44
Training error	0.031	0.027	0.026	0.026
Testing error	0.031	0.028	0.027	0.027
Iterations	8,700	2,300	2,900	3,800

In summary, this example illustrates a systematic way of using PCA to ease the use of IT-nets for a large data set. As a first step, observed data can be preprocessed using PCA to reduce linearly correlated variables. The performance of PCA can help estimate the number of input nodes for the IT-net. The results of PCA also provide an initial guess of input values for training the IT-net.

Note that in our approach to reducing data dimensionality, values for all latent variables are computed simultaneously. This approach differs from the sequential nonlinear principal component analysis (NLPCA) suggested by Kramer (1991) and adopted by Dong and McAvoy (1993). In the sequential NLPCA, one computes the first nonlinear PC from the observed data, then the second nonlinear PC from the residuals of the previous computation, and so forth.

It should be pointed out that the sequential method is not equivalent to the simultaneous one when a nonlinear model is involved. When two latent variables are needed to reproduce observed variables, a general nonlinear model for dimensionality reduction should be:

$$t_k = \phi_k(\lambda_1, \lambda_2) + e_k, \quad k = 1, \ldots, n.$$

The sequential procedure for computing principal components actually assumes that each of the two-variable functions, $\phi_1, \phi_2, \ldots, \phi_n$, is the sum of two single-variable functions:

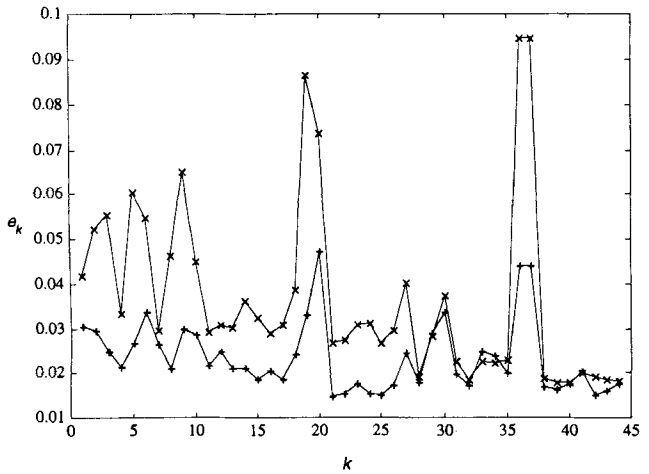

Figure 13. Root-mean-square errors for variables.

× = PCA with 3 PCs; + = 3-33-44 IT-net.

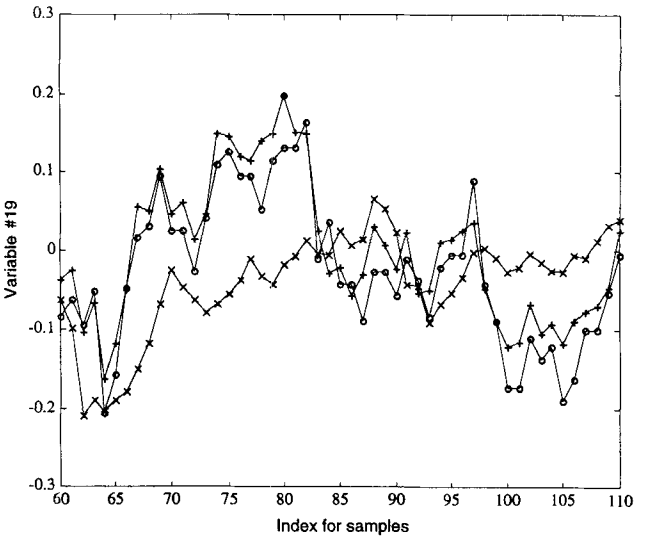

Figure 14. Original data and reproduced values for variable 19.

○ = original data; + = 3-33-44 IT-net; × = PCA with 3 PCs.

$$\phi_k(\lambda_1, \lambda_2) = \psi_{k1}(\lambda_1) + \psi_{k2}(\lambda_2).$$

These functions only represent a very limited class of nonlinear models. Many commonly used nonlinear functions do not fall into this class. For example,

$$\phi_k(\lambda_1, \lambda_2) = \lambda_1 \log(\lambda_1 + \lambda_2).$$

Therefore, the sequential method may limit the types of nonlinear relationships that can be captured through dimensionality reduction, whereas the simultaneous method is able to make use of any nonlinear models.

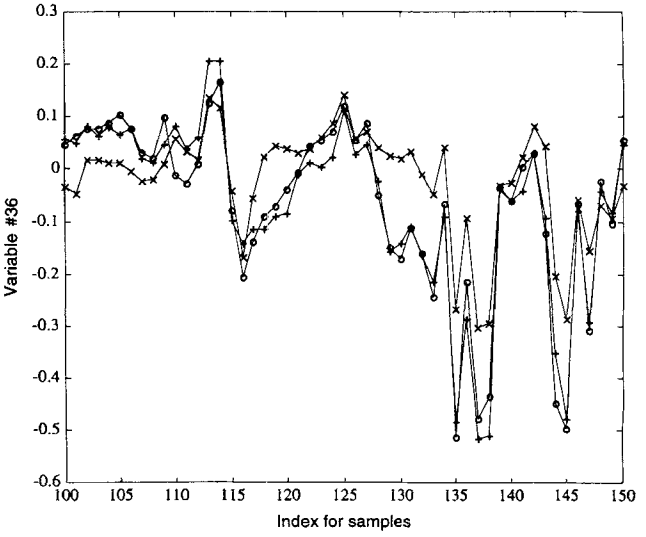

Figure 15. Original data and reproduced values for variable 36.

○ = original data; + = 3-33-44 IT-net; × = PCA with 3 PCs.

Conclusions

Reduction of data dimensionality is an important step toward understanding complicated chemical processes. By introducing the concept of input training, we have developed a new type of network, called IT-net, for reducing data dimensionality. The IT-net approach works by extending backpropagation training to network inputs. An IT-net requires only one hidden layer to capture nonlinear correlations in observed data and hence offers better results with less computational effort than an autoassociative network, which typically needs three hidden layers.

The IT-net approach is close in spirit to principal curves in that it alternates between improving model parameters for given inputs and finding the optimal inputs for given model parameters. An advantage of the IT-net approach is its association with neural computing. Using IT-nets for nonlinear PCA is analogous to using ordinary feedforward networks for nonlinear regression. Much of the knowledge on constructing and training backpropagation networks can be applied to IT-nets for better performance.

Through examples we show that IT-nets are capable of reducing random and systematic errors in data sets by exploiting correlations among variables. A properly trained IT-net is capable of providing model-consistent estimates for data containing random or systematic errors. Numerical experiments also show that training an IT-net on noisy data reduces random errors while retaining all nonrandom variations.

A relatively large IT-net has been successfully applied to a set of real measurement data from an electric power plant. Through the application, we used a systematic way of exploiting PCA in the IT-net approach. PCA provides both a guideline for selecting the number of input nodes and an initial guess of input values for training the IT-net. The IT-net approach yields substantially better results for the data set than PCA.

IT-nets can be applied to the same problems as autoassociative networks: reducing data noise, estimating missing measurements, detecting and correcting gross errors. Latent variables can be used for assessing process performance and for diagnosis. We are currently investigating the application of this approach to a data set involving more processing units of different types, combining IT-nets with a hierarchical decomposition of the set of variables (Mavrovouniotis and Chang, 1992).

Acknowledgments

This project is funded by Electric Power Research Institute, EPRI grant number RP8017-4 (Dr. Martin Wildberger). The authors thank Prof. Thomas J. McAvoy for an advance draft of his paper and for pointing out the relevance of principal curves.

Notation

E = sum of square errors
i = index for input nodes, or latent variables
j = index for hidden nodes
k = index for output nodes, or observed variables
p = index for data samples
x = vector of input values
x_{pi} = input value of input node i for sample p
λ_i = ith input variable
ϕ_k = function of the kth output node

Literature Cited

Abbas, H. M., and M. M. Fahmy, "Neural Model for Karhunen Loeve Transformation with Application to Adaptive Image Compression," *Proc. Inst. Elec. Eng., I. Commun. Speech Vision*, **2**, 135 (1993).

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Sci.*, **9**, 147 (1985).

Carroll, D. J., "Polynomial Factor Analysis," *Proc. Annu. Convention Amer. Psychological Assoc.*, Arlington, VA, p. 103 (1969).

Cottrel, G. W., P. Munro, and D. Zipser, "Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming," *Proc. Conf. Cognitive Sci. Soc.*, p. 461 (1987).

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. Control, Signals, Syst.*, **2**, 303 (1989).

Dong, D., and T. J. McAvoy, "Nonlinear Principal Component Analysis Based on Principal Curves and Neural Networks," *Comput. Chem. Eng.*, submitted (1994).

Hastie, T., and W. Stuetzle, "Principal Curves," *J. Amer. Stat. Assoc.*, **84**, 502 (1989).

Hertz, J., A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, CA, p. 198 (1991).

Jolliffe, I. T., *Principal Component Analysis*, Springer-Verlag, New York, p. 122 (1986).

Kramer, M. A., "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE J.*, **37**, 233 (1991).

Kramer, M. A., "Autoassociative Neural Networks," *Compu. Chem. Eng.*, **16**, 313 (1992).

MacGregor, J., "Multivariate Statistical Methods for Monitoring Large Data Sets from Chemical Processes," paper 164a, AIChE Meeting, San Francisco (1989).

Mavrovouniotis, M. L., and S. Chang, "Hierarchical Neural Networks," *Compu. Chem. Eng.*, **16**, 347 (1992).

McDonald, R. P., *Nonlinear Factor Analysis*, William Byrd, Richmond, VA (1967).

Piovoso, M. J., K. A. Kosanovich, and J. P. Yuk, "Process Data Chemometrics," *IEEE Trans. Instrum. Meas.*, **IM2**, 262 (1992).

Raich, A., and A. Cinar, "Statistical Process Monitoring and Disturbance Isolation," *Proc. Midwest Process Control Workshop*, West Lafayette, IN (1994).

Stephanopoulos, G. N., and H. Guterman, "Pattern Recognition in Fermentation Processes," paper 163, Amer. Chem. Soc. Meeting, Miami Beach, FL (1989).

Strang, G., *Linear Algebra and Applications*, Academic Press, New York (1988).

Wise, B. M., and N. L. Ricker, "Upset and Sensor Failure Detection in Multivariate Processes," paper 164b, AIChE Meeting, San Francisco (1989).

Manuscript received Mar. 2, 1994, and revision received Aug. 1, 1994.