Nonlinear Principal Component Analysis Using Autoassociative Neural Networks

Mark A. Kramer
Laboratory for Intelligent Systems in Process Engineering, Dept. of Chemical Engineering,
Massachusetts Institute of Technology, Cambridge, MA 02139

Nonlinear principal component analysis is a novel technique for multivariate data analysis, similar to the well-known method of principal component analysis. NLPCA, like PCA, is used to identify and remove correlations among problem variables as an aid to dimensionality reduction, visualization, and exploratory data analysis. While PCA identifies only linear correlations between variables, NLPCA uncovers both linear and nonlinear correlations, without restriction on the character of the nonlinearities present in the data. NLPCA operates by training a feedforward neural network to perform the identity mapping, where the network inputs are reproduced at the output layer. The network contains an internal "bottleneck" layer (containing fewer nodes than input or output layers), which forces the network to develop a compact representation of the input data, and two additional hidden layers. The NLPCA method is demonstrated using time-dependent, simulated batch reaction data. Results show that NLPCA successfully reduces dimensionality and produces a feature space map resembling the actual distribution of the underlying system parameters.

Introduction

Engineers are often confronted with the problem of extracting information about poorly-known processes from data. Discerning the significant patterns in data, as a first step to process understanding, can be greatly facilitated by reducing dimensionality. The *superficial dimensionality* of data, or the number of individual observations constituting one measurement vector, is often much greater than the *intrinsic dimensionality*, the number of independent variables underlying the significant nonrandom variations in the observations. The reduction of a data set from its superficial to intrinsic dimensions is the focus of this article.

The problem of dimensionality reduction is closely related to *feature extraction*. Feature extraction refers to identifying the salient aspects or properties of data to facilitate its use in a subsequent task, such as regression or classification (Duda and Hart, 1973). Its features are a set of derived variables, functions of the original problem variables, which efficiently capture the information contained in the original data. For example, a useful feature for recognizing a leak in a processing unit is the difference between the influent and effluent flow rates. Using the net flow rate to identify the leak simplifies the identification task (since a leak is only possible when the net flow is nonzero) and reduces the dimensionality of the recognition problem by substituting a single indicator for two or more individual measurements. In general, selection of features such as the net flow rate depends on the ultimate application. However, in cases where the ultimate application of the data is not known in advance, a suitable objective for feature extraction is to reduce dimensionality with minimal loss of information (Földiák, 1989). The nonlinear principal component analysis (NLPCA) introduced here is a general-purpose feature extraction algorithm producing features that retain the maximum possible amount of information from the original data set, for a given degree of data compression. Information preservation assures that the selected features will be useful in most situations, independent of the ultimate application.

Within the class of linear methods, the optimal information-preserving transformation is given by principal component analysis (PCA) (Fukunaga and Koontz, 1970). Reduction of dimensionality by PCA has been shown to facilitate many types of multivariate analysis, including data validation and fault detection (Wise and Ricker, 1989), quality control (MacGregor, 1989), correlation and prediction (Joback, 1984), and data visualization (Stephanopoulos and Guterman, 1989). The feature variables in PCA, also called *factors*, are linear combinations of the original problem variables. The coefficients of the linear transformation are such that if the feature transformation is applied to the data set and then reversed, there will be a minimum sum of squares difference between the original and reconstructed data. The same criterion of optimality is adopted in NLPCA. The main difference between PCA and NLPCA is that the latter involves nonlinear mappings between the original and reduced dimension spaces. If nonlinear correlations between variables exist, NLPCA will describe the data with greater accuracy and/or by fewer factors than PCA, provided that there are sufficient data to support the formulation of more complex mapping functions.

The NLPCA method uses artificial neural network (ANN) training procedures to generate nonlinear features. The networks are of a conventional type, featuring feedforward connections and linear or sigmoidal nodal transfer functions, trained by backpropagation. The particular network architecture used employs three hidden layers, including an internal "bottleneck" layer of smaller dimension than either input or output. The network is trained to perform the identity mapping, where the input is approximated at the output layer. Since there are fewer units in the bottleneck layer than the output, the bottleneck nodes must represent or encode the information in the inputs for the subsequent layers to reconstruct the input (Sanger, 1989). If network training finds an acceptable solution, a good representation of the input must exist in the bottleneck layer. This implies that data compression caused by the network bottleneck may force hidden units to represent significant features in data. The concept of using a neural network with a bottleneck to concentrate information has been previously discussed in the context of "encoder/decoder" problems (Ackley et al., 1985; Rumelhart et al., 1986; Sanger, 1989). However, it will be shown that the previous network architectures used for encoder/decoder problems do not provide optimal nonlinear feature extraction because of limitations in the representational capacity of the networks. The current work provides a new network architecture that overcomes the previous limitations.

In the following section, PCA is reviewed as background for the current developments. The NLPCA technique is then developed. Artificial neural networks are not reviewed here; the reader is referred to Rumelhart et al. (1986), Hoskins and Himmelblau (1988), or Venkatasubramanian and Chan (1989) for background information. Finally, NLPCA is applied to test problems, and results comparing PCA and NLPCA are presented.

Principal Component Analysis

PCA is a technique for mapping multidimensional data into lower dimensions with minimal loss of information. Since PCA has been described frequently by other authors (e.g., Mardia et al., 1980), only a brief summary is given here. Let $\underline{\underline{Y}}$ represent a $n \times m$ table of data (n = number of observations, m = number of variables). PCA is an optimal factorization of $\underline{\underline{Y}}$ into two matrices, $\underline{\underline{T}}$ called the *scores* matrix ($n \times f$, and $\underline{\underline{P}}$, called the *loadings* matrix ($m \times f$), plus a matrix of residuals $\underline{\underline{E}}$ ($n \times m$):

$$\underline{\underline{Y}} = \underline{\underline{T}}\,\underline{\underline{P}}^T + \underline{\underline{E}} \tag{1}$$

where f is the number of factors ($f < m$). The condition of optimality on the factorization is that the Euclidean norm of the residual matrix, $\|\underline{\underline{E}}\|$, must be minimized for the given number of factors. To satisfy this criterion, it is known that the columns of $\underline{\underline{P}}$ are the eigenvectors corresponding to the f largest eigenvalues of the covariance matrix of $\underline{\underline{Y}}$.

It is useful to view PCA as a linear mapping of data from $\mathbb{R}^m$ to $\mathbb{R}^f$. Taking $\underline{\underline{P}}^T\underline{\underline{P}} = \underline{\underline{I}}$ without loss of generality, the mapping has the form:

$$\underline{T} = \underline{Y}\,\underline{\underline{P}} \tag{2}$$

where $\underline{Y}$ represents a row of $\underline{\underline{Y}}$, a single data vector, and $\underline{T}$ represents the corresponding row of $\underline{\underline{T}}$, or the coordinates of $\underline{Y}$ in the feature space. The loadings $\underline{\underline{P}}$ are the coefficients for the linear transformation. The information lost in this mapping can be assessed by reconstruction of the measurement vector by reversing the projection back to $\mathbb{R}^m$:

$$\underline{Y}' = \underline{T}\,\underline{\underline{P}}^T \tag{3}$$

where $\underline{Y}' = \underline{Y} - \underline{E}$ is the reconstructed measurement vector. The smaller the dimension of the feature space, the greater the resulting error, measured by $|\underline{E}|$ for individual measurement vectors, or $\|\underline{\underline{E}}\|$ for the overall data set.

Nonlinear Principal Component Analysis

In NLPCA, the mapping into feature space is generalized to allow arbitrary nonlinear functionalities. By analogy to Eq. 2, we seek a mapping in the form:

$$\underline{T} = \underline{G}(\underline{Y}) \tag{4}$$

where $\underline{G}$ is a nonlinear vector function, composed of f individual nonlinear functions; $\underline{G} = \{G_1, G_2, \ldots, G_f\}$, analogous to the columns of $\underline{\underline{P}}$, such that if T_i represents the ith element of $\underline{T}$,

$$T_i = G_i(\underline{Y}) \tag{5}$$

By analogy to the linear case, G_1 is referred to as the primary nonlinear factor, and G_i is the ith nonlinear factor of $\underline{\underline{Y}}$.

The inverse transformation, restoring the original dimensionality of the data, analogous to Eq. 3, is implemented by a second nonlinear vector function $\underline{H} = \{H_1, H_2, \ldots H_m\}$:

$$Y_j' = H_j(\underline{T}) \tag{6}$$

The loss of information is again measured by $\underline{E} = \underline{Y} - \underline{Y}'$, and analogous to PCA, the functions $\underline{G}$ and $\underline{H}$ are selected to minimize $\|\underline{\underline{E}}\|$.

To generate $\underline{G}$ and $\underline{H}$, a basis function approach is used. Cybenko (1989) has shown that functions of the following form are capable of fitting any nonlinear function $\underline{v} = f(\underline{u})$ to an arbitrary degree of precision:

$$v_k = \sum_{j=1}^{N_2} w_{jk2}\,\sigma\left(\sum_{i=1}^{N_1} w_{ij1} u_i + \theta_{j1}\right) \tag{7}$$

where $\sigma(x)$ is any continuous and monotonically increasing function with $\sigma(x) \rightarrow 1$ as $x \rightarrow +\infty$ and $\sigma(x) \rightarrow 0$ as $x \rightarrow -\infty$. A suitable function is the sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{8}$$

Equations 7-8 are the describing equations for a feedforward artificial neural network (ANN) with N_1 inputs, a hidden layer comprised of N_2 nodes with sigmoidal transfer functions, and a linear output node (implementing a summation of its inputs) for each k. In Eq. 7, w_{ijk} represents the weight on the connection from node i in layer k to node j in layer $k+1$. Nodes in successive layers are exhaustively interconnected, and there are no intralayer links. The θ are nodal biases, treated as adjustable parameters like the weights. More details on neural networks of this type are given in Rumelhart et al. (1986). Note that to achieve the universal fitting property, exactly one layer of sigmoidal nodes and two layers of weighted connections are required. In practice, sigmoidal nonlinearities are often included in the nodes of the output layer so that the network produces outputs in a fixed, finite range. Also, the sigmoid function can be scaled multiplicatively or translated without affecting the generality of the network. Since we frequently deal with mean-centered data sets, a sigmoid function scaled into the range $(-1,1)$ was used in this work.

The ability of the neural network to fit arbitrary nonlinear functions depends on the presence of a hidden layer with nonlinear nodes. Without the hidden layer (or with linear nodes in the hidden layer), the network is only capable of producing linear combinations of the inputs, given linear nodes in the output layer. A network lacking a hidden layer but including sigmoidal nonlinearities in the output layer is only capable of generating multivariable sigmoidal functions, i.e., linear functions compressed into the range $(-1,1)$ by the sigmoid. Neither network without a hidden layer is compatible with the goal of representing arbitrary nonlinear functions. Therefore, the architecture for the networks representing $\underline{G}$ and $\underline{H}$ are as follows. The network for $\underline{G}$ operates on the rows of $\underline{\underline{Y}}$ and has m inputs. The hidden layer of $\underline{G}$, which we call the *mapping* layer, contains M_1 nodes with sigmoidal transfer functions, where $M_1 > f$. The output of the network is the projection of the input vector into feature space, and therefore contains f nodes. The output nodes can have linear or sigmoidal transfer functions, without affecting the generality of $\underline{G}$. The function G_i, the ith nonlinear factor, is defined by the weights and biases on the connections from the input to the ith output. The network representing the inverse mapping function $\underline{H}$ takes the rows of $\underline{\underline{T}}$ as inputs and accordingly has f inputs. The hidden layer, which we refer to as the *demapping* layer, contains M_2 nodes with sigmoidal transfer functions, where $M_2 > f$. The output layer yields the reconstructed data, $\underline{Y}'$, and thus contains m nodes. The nodes of the output layer can be linear or sigmoidal. The weights and biases connecting the inputs to the ith output node define the function H_i. These network architectures for modeling $\underline{G}$ and $\underline{H}$ are shown in Figure 1.

ANNs of the type used here require "supervised" training, where a desired output is specified for each training example. However, in training the network representing $\underline{G}$, the desired output $\underline{T}$ is unknown. For the network representing $\underline{H}$, the desired outputs are known (the target output is $\underline{Y}$), but the corresponding inputs $\underline{T}$ are not. Therefore, direct supervised training of these networks is not feasible. To circumvent this problem, it is observed that $\underline{T}$ is both the output of $\underline{G}$ and the input of $\underline{H}$. Combining the two networks in series so that $\underline{G}$ feeds directly into $\underline{H}$, a network is obtained whose inputs and desired outputs are known (Figure 2). Specifically, $\underline{Y}$ is both the input to $\underline{G}$ and the desired output from $\underline{H}$; thus, the combined network must be trained to produce the identity mapping, $\underline{Y} \rightarrow \underline{Y}$. Supervised training can thus be applied to the combined network. Training to learn the identity mapping has been called *self-supervised backpropagation* or *autoassociation* (Ballard, 1987; Cottrell et al., 1987), and has been used to solve so-called "encoder/decoder" problems, discussed in the subsection on Relationship to Other Work.

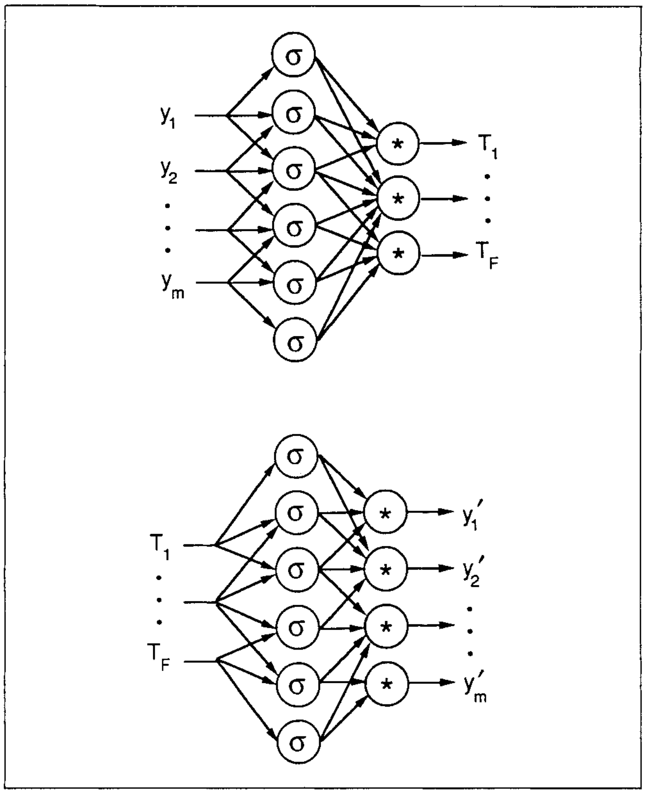


Figure 1. Networks implementing mapping and demapping functions.

Top, network for $\underline{G}$; bottom, network for $\underline{H}$
σ indicates sigmoidal nodes; $*$ indicates linear or sigmoidal nodes.

The combined network shown in Figure 2 contains *three hidden layers*, the mapping layer involved in modeling in $\underline{G}$, the middle layer whose outputs represent the features $\underline{T}$, and the demapping layer involved in modeling $\underline{H}$. We refer to the second hidden layer of the combined network as the *bottleneck layer* because it is the smallest in dimensionality. The input and output layers of the combined network represent $\underline{Y}$ and $\underline{Y}'$, respectively. Note that the nodes of the mapping and demapping layers must have nonlinear transfer functions to provide the capability for modeling arbitrary $\underline{G}$ and $\underline{H}$. However, nonlinear nodes are *not* required in the bottleneck layer, since the bottleneck represents the output layer of subnet $\underline{G}$. Nonetheless, if a bounded response in the feature space is desired, sigmoids can be used in all hidden layers. If desired, the network can also include linear "bypass" connections from

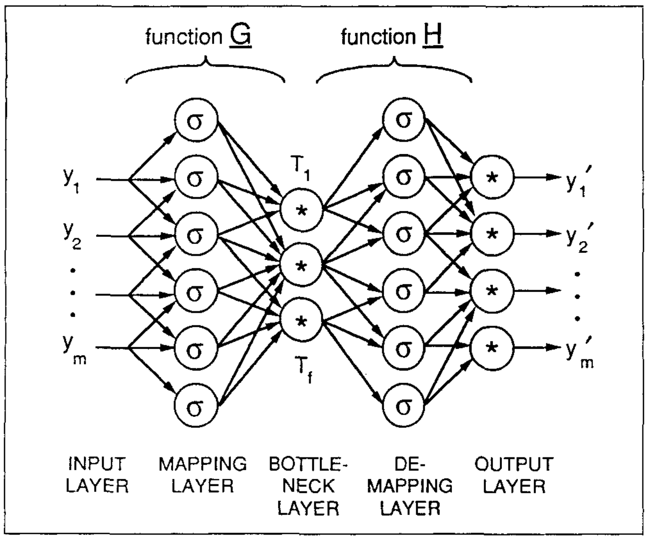


Figure 2. Network architecture for simultaneous determination of *f* nonlinear factors using an autoassociative network.

σ indicates sigmoidal nodes, $*$ indicates sigmoidal or linear nodes.

the input directly to the bottleneck layer and from the bottleneck layer to the output, but not across the bottleneck. Linear bypasses can increase the ease with which the network learns linear functionalities (Haesloop and Holt, 1990). The networks presented here do not include linear bypasses.

The reader may note that if the bottleneck layer consists of linear nodes, then the weights of the pre- and post-bottleneck layers can be combined to create a network containing only two hidden layers. Specifically, if $\underline{\underline{W}}_2$ and $\underline{\underline{W}}_3$ represent matrices of weights in the layers directly before and after the bottleneck, then $\underline{\underline{W}}_{23} = \underline{\underline{W}}_2 \underline{\underline{W}}_3$ represents equivalent weights for direct connection of the mapping and demapping layers. It is tempting to conclude that NLPCA could be implemented by training a network consisting of only two hidden layers, omitting the bottleneck. However, this would not be the same as training the three hidden layer network. Since $\underline{\underline{W}}_2$ is dimension M_1 by f, and $\underline{\underline{W}}_3$ is dimension f by M_2, the rank of $\underline{\underline{W}}_{23}$ is f, even though its superficial dimension is M_1 by M_2. If a two-hidden layer network were trained in the manner discussed above, the required deficiency in rank would not develop. Thus, there is a subtle but crucial difference between a network with two hidden layers, and a network containing three hidden layers where the second hidden layer is a bottleneck with linear nodes.

To further appreciate the requirement for three hidden layers, consider the implication of eliminating the mapping and demapping layers. This would leave a combined network with only one hidden layer, the bottleneck layer between inputs and outputs. If the nodes of the bottleneck layer were linear, this would correspond exactly to (linear) PCA, as shown by Sanger (1989). If the bottleneck nodes were sigmoidal, the functional forms of $\underline{G}$ and $\underline{H}$ would still be severely constrained; only linear combinations of the inputs compressed by the sigmoid into the range $(-1,1)$ could be represented. Therefore, the performance of an autoassociative network with only one internal layer of sigmoidal nodes is often no better than linear PCA. This is demonstrated in the example problems.

These considerations lead to the conclusion that to achieve optimal nonlinear feature extraction in the general case, three hidden layers are essential. This architecture comes from combining the networks for the two nonlinear functions $\underline{G}$ and $\underline{H}$ in series. Each individual function is nonlinear and therefore requires a hidden layer of nonlinear nodes. Along with the shared internal input-output layer that serves as the bottleneck, a network with three hidden layers results. The result of Cybenko that one hidden layer is sufficient to model any nonlinear function (with enough hidden nodes) applies to the functions $\underline{G}$ and $\underline{H}$ individually, not to the combined network as a whole.

To train the combined network, the weights appearing in the networks representing $\underline{G}$ and $\underline{H}$ are optimized so that the reconstructed outputs $\underline{\underline{Y}}'$ match the inputs $\underline{\underline{Y}}$ as closely as possible. Training is complete when E, the sum of squared errors given in Eq. 9, is minimized:

$$E = \sum_{p=1}^{n} \sum_{i=1}^{m} (Y_i - Y_i')_p^2 \tag{9}$$

E is the square of $\|\underline{\underline{E}}\|$, the optimality criterion used in PCA. Therefore, minimizing E during network training results in minimum information loss in the same sense as PCA. The actual training can be approached by any appropriate algorithm such as backpropagation (Werbos, 1974; Rumelhart et al., 1986). In this work, a conjugate gradient version of backpropagation with better speed and convergence properties was used (Leonard and Kramer, 1990).

After training, the combined network no longer has any utility, and it is disaggregated into the two single-hidden layer networks representing $\underline{G}$ and $\underline{H}$. In most applications, $\underline{G}$ is the function of interest since it carries out the feature space mapping. The data are propagated through $\underline{G}$ to project the data into low-dimensional feature space.

Selection of mapping nodes

In the combined network, there are m nodes in the input and output layers and f nodes in the bottleneck layer. However, there is no definitive method for deciding *a priori* the dimensions of the mapping and demapping layers (henceforth, collectively referred to as the mapping layers). The number of mapping nodes is related to the complexity of the nonlinear functions that can be generated by the network. If too few mapping nodes are provided, accuracy might be low because the representational capacity of the network is limited. However, if there are too many mapping nodes, the network will be prone to "overfitting," that is, learning the stochastic variations in the data rather than the underlying functions. The simplest approach to this problem is to restrict the number of weights in the network to a fraction of the number of constraints imposed by the data set. For each data vector, a separate constraint is imposed by each output node, so that the total number of adjustable parameters must be less than $n \cdot m$. For the combined architecture, assuming all nodes have biases, the number of adjustable parameters in the network is $(m+f+1)(M_1+M_2)+m+f$, where M_1 and M_2 are the number of nodes in the mapping and demapping layers, respectively, implying the inequality:

$$M_1+M_2 \ll m(n-f)/(m+f+1) \quad (10)$$

For a relatively small number of factors ($f \ll m, n$), Eq. 10 is approximated by:

$$M_1+M_2 \ll n \quad (11)$$

If the number of mapping and demapping nodes allowed by these inequalities is less than or equal to f, then there is not enough data to support extraction of f nonlinear factors, since the bottleneck by design occurs in the second hidden layer of the combined network.

Two other approaches for selecting the number of mapping nodes are worth mentioning. The first method involves randomly splitting the data set into two subsets, using one of the sets for training, and the other for testing the generality of the result. If the average error on the training subset is significantly less than on the testing subset (say by a factor of 2 or more), overfitting is indicated, and the dimension of the mapping layers should be reduced. The second method involves the use of functions that express the trade-off between fitting accuracy and number of adjustable parameters in explicit terms. Two such functions are Akaike's criteria, the final prediction error (FPE), and information theoretic criterion (AIC) (Ljung, 1987):

$$\text{FPE}=e(1+N_w/N)/(1-N_w/N) \quad (12)$$

$$\text{AIC}=\ln(e)+2N_w/N \quad (13)$$

Here, $N_w=(m+f+1)(M_1+M_2)+m+f$ is the number of weights, $N=nm$ is the number of entries in the data matrix, and $e=E/(2N)$ is an average sum of squares error. Minimization of these functions identifies models that are neither over- nor underparameterized. For $N_w \ll N$, AIC and FPE are approximately the same, but for larger N_w, FPE will tend to increase faster, indicating a preference for models with fewer adjustable parameters. To apply Eqs. 12–13, several training runs with different mapping layer dimensions are needed to determine the location of the minima in FPE and AIC. Because these criteria can be used to compare models of different structure, they can also be used to compare PCA and NLPCA, as explained in the Example section.

Sequential NLPCA

Two different approaches have been used to carry out NLPCA. The first approach solves the problem as depicted in Figure 2, that is, the bottleneck layer of the network is provided with as many nodes as the total number of nonlinear factors desired, and the network is trained to produce the identity mapping using the objective function of Eq. 9. For $f>1$, an alternate serial training procedure can be used. This procedure is similar in spirit to the recursive procedure of factor calculation often used in PCA. This method is based on the fact that the primary factor of the residual matrix $\underline{\underline{E}}$ is the second factor of the original data matrix. Applied recursively, this implies that if $\underline{p}_i(\underline{\underline{Y}}_1)$ is the ith column of $\underline{\underline{P}}$ (the ith factor) for a given data matrix $\underline{\underline{Y}}_1$, then:

$$\underline{p}_i(\underline{\underline{Y}}_1)=\underline{p}_{i-1}(\underline{\underline{Y}}_2)=\ldots=\underline{p}_1(\underline{\underline{Y}}_i) \quad (14)$$

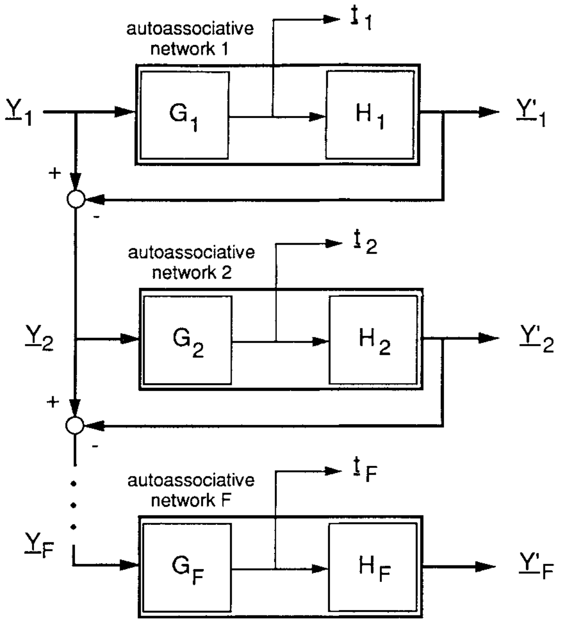


Figure 3. Sequential determination of nonlinear factors by training *F* networks with one bottleneck node each.

where $\underline{\underline{E}}_i=\underline{\underline{Y}}_{i+1}$ is the residual matrix generated after extraction of the primary factor from matrix $\underline{\underline{Y}}_i$. Various algorithms, for example, NIPALS (Geladi and Kowalski, 1986), can calculate the primary factor $\underline{p}_1$ of a given matrix efficiently. Used recursively according to Eq. 14, multiple factors of the original data matrix can be found.

The serial NLPCA algorithm is similar in spirit. A series of F separate networks, each containing a *single* node in the bottleneck layer, are trained. Only the primary nonlinear factor of the input matrix is extracted at each stage. The residual from the previous network becomes the input (and target output) for the succeeding network, and the training proceeds serially. The recursion relations for this process are:

$$\begin{aligned} \underline{t}_i &= G_1(\underline{\underline{Y}}_i) \\ \underline{\underline{Y}}_{i+1} &= \underline{\underline{Y}}_i-\underline{\underline{Y}}_i{}'=\underline{\underline{Y}}_i-H_i(\underline{t}_i) \end{aligned} \quad (15)$$

where $\underline{t}_i$ is the ith column of $\underline{\underline{T}}$. The network architecture for the sequential calculation is shown in Figure 3.

The serial version of NLPCA has several potential advantages over the simultaneous method of factor determination. First, the serial training procedure allows rescaling of the residual matrix between steps, which may hasten convergence. Since the residuals get progressively smaller, it may be difficult for the network to model the higher factors without rescaling. Second, the serial training forces each bottleneck node to model a separate factor in the data. With simultaneous training of two or more bottleneck nodes, we have observed a tendency for two or more of the bottleneck nodes to align with the primary factor early in the training, when all the weight changes respond to the largest output errors. If the weights into the bottleneck layer come into alignment, there is an effective reduction in the number of bottleneck nodes. The primary factor may thus appropriate the representational resources intended for secondary and later factors, leaving the network unable to move further toward the global optimum. This was noted in one example, not reported in this article. We have not yet undertaken a comprehensive comparison to recommend either serial or simultaneous versions of NLPCA. The examples were done with simultaneous extraction of factors, and no problems were encountered.

Relationship to other work

One of the first papers linking PCA, feature extraction and neural networks was by Oja (1982). Oja showed that PCA can be carried out by a linear (summing) neuron whose input connection strengths vary in time according to a Hebbian-type learning law:

$$w_i(t+1) = w_i(t) + \gamma\eta(t)[Y_i(t) - \eta(t)w_i(t)] \tag{16}$$

where γ is a positive learning rate, w_i represents the connection weight of the ith input, $Y_i(t)$ is a stochastic variable introduced on the ith connection, and $\eta(t) = \underline{w}(t)\underline{Y}(t)$ is the neuron output. For small γ, Oja shows the weights $\underline{w}$ converge to the primary factor of the inputs, and the node output $\eta(t)$ to the score of the input vector $\underline{Y}(t)$. This method was extended by Földiák (1989) to networks that calculate more than one principal component. Sanger (1989) proposes a similar strategy for PCA in linear networks, based on self-supervised backpropagation learning. In a related work, Baldi and Hornik (1989) prove that self-supervised backpropagation in linear networks has a unique minimum at values of weights corresponding to the principal components of the inputs. Therefore, Hebbian and backpropagation learning both lead to the PCA solution in linear networks. These works focus on linear networks and linear features, unlike the current presentation on nonlinear feature extraction.

Kohonen (1984) maps high-dimensional data to two dimensions using a network architecture called a self-organizing feature map. The data are clustered by a sequential algorithm similar to k-means clustering (Duda and Hart, 1973). Each cluster center contains internal parameters representing a location in the input space and is given a location in a two-dimensional grid. Neighbors in the grid represent centers closest to one another in the original space. A new input excites the cluster center nearest to it in the original space, and the location of the excited center in the grid is indicative of the location of the input in the high-dimensional space. This and other methods of planar mapping (Siedlecki et al., 1988) have the disadvantage that all data sets, regardless of their intrinsic dimensionality, are represented in two dimensions. Thus, the user has no control over the loss of information incurred in the mapping process.

The ability of nonlinear ANNs to develop abstract internal representations in their hidden layers was discussed by Hinton (1986). Use of this ability to derive compact representations using autoassociative mapping and a bottleneck layer appears to have been demonstrated first by Ackley et al. (1985) in the context of the "encoder/decoder" problem. In this problem, the network is presented with M distinct binary input patterns consisting of M bits each, each with one bit on (set to 1) and the remaining bits off (set to 0). The task is to concentrate the input information into $\log_2 M$ bits represented by the nodes of the hidden layer (encoding) and, based on the state of the hidden layer, turn on the same bits in the output layer as presented at the input layer (decoding). This has been a popular test problem for network training methodologies (Rumelhart et al., 1986; Fahlman, 1989). For studying feature development, however, the binary encoder is an unfortunate choice, because unlike other autoassociation problems, one hidden layer is sufficient to solve it (in this case $\underline{G}$ and $\underline{H}$ are composed of "and" and "or" functions, implementable in a single layer each). Autoassociative networks with one hidden layer have been applied to the compression of grey-scale images by Cottrell et al. (1987), and Chauvin (1989) applies an architecture with two hidden layers.

Ballard (1987) uses multiple single-hidden layer networks to build complex encodings. The first autoassociative network is trained to encode the original data set. The output of the hidden layer from the first network is then used as training data for a second autoassociative network, which by encoding the internal representation of the first network builds a more abstract encoding. Although Ballard experiments only with two levels of encoding, in theory the process can be repeated recursively to generate higher-order abstractions. For two levels of recursion, Ballard's architecture is equivalent to a three-hidden-layer autoassociative network. However, the training scheme imposes a constraint equivalent to requiring equality between the output of the mapping layer and the output of the demapping layer in NLPCA. Since this constraint restricts the mapping and demapping functions, Ballard's architecture does not appear to have the capability of modeling arbitrary nonlinear features. A similar criticism seems to apply to the network of Miikkulainen and Dyer (1989), which employs an adaptive external "lexicon" as the source of examples to a single-hidden layer autoassociative network. Due to the specialized architectures of these networks, they do not appear to be capable of general nonlinear feature extraction.

In some previous reports, techniques for dynamic modification of the network architecture during training to create "skeleton networks," containing only essential connections, have been proposed. If there is redundancy in the inputs, skeletonization methods can discover the minimal set of inputs needed to make the desired output predictions through selective removal of unnecessary connections and nodes. Krusche (1989) introduces a methodology for simultaneous training and network bottlenecking that gradually excises nodes whose input weight vectors are nearly parallel through internal competition. Mozer and Smolensky (1989) reduce ANN complexity by calculating an "attentional strength coefficient" for each input and hidden layer node. They propose eliminating the node with the least attentional strength, retraining the reduced net and repeating the process until the desired network reduction is achieved. Karnin (1990) estimates the sensitivity of the error to the exclusion of each connection and eliminates the connections with lowest sensitivities. Niida et al. (1989) achieve a similar goal by modifying the usual training objective function with a term that penalizes nonzero weights. Weights that become smaller than a predetermined threshold value during training are eliminated. Input nodes that become isolated by zero weights during the training process are either redundant or exert a minimal influence on the outputs.

Removal of redundant inputs is reminiscent of NLPCA. However, skeletonization methods require a subset of the vari-

variables to be specified in advance as outputs, which are then calculated as functions of the remaining variables. If skeletonization methods were applied to autoassociative networks, it is almost inevitable that "tautologous" networks would result, where all cross-connections between input i and output j for $i \neq j$ are eliminated, and only "straight-through" connections retained. This would reduce the number of weights, but would not concentrate the input information. In addition, most skeletonization methods allow the user only indirect control over the extent of network reduction through parameters that weight the trade-off between fitting error and network complexity. By specifying the bottleneck dimension in advance, NLPCA avoids producing tautologous networks and allows direct control over the number of factors calculated from the data. Skeletonization methods might enhance NLPCA if the initial network architecture already contains a bottleneck. In this case, the mapping and demapping layers would be the primary targets of skeletonization.

Examples

Examples of NLPCA with comparisons to PCA are presented in this section. Networks for the NLPCA method were trained using the conjugate gradient version of backpropagation discussed by Leonard and Kramer (1990). Since the conjugate gradient method dynamically optimizes the learning rate and momentum factor in the backpropagation learning law, these did not enter as study parameters. In addition, the number of iterations are not reported since iterations depend on the training method and the initial conditions, although in most cases less than 1,000 iterations were needed to converge. In all cases, training was continued until there was no more progress in reducing E (fractional step-to-step change less than 10^{-6}). Training was repeated several times with different random initial conditions for each architecture to confirm that the global minimum had been found. The dimensions of the mapping layers are varied in these experiments, but for simplicity all networks tested had $M_1 = M_2$, which is not a requirement of the method. Also, sigmoidal nodes were used in all hidden layers as well as the output layer so that the factors and outputs were bounded in the range $(-1,1)$.

Example 1

A simple test of the NLPCA methodology is provided by a data set consisting of two observed variables y_1 and y_2, driven by one underlying parameter, θ:

$$y_1 = 0.8\sin\theta$$

$$y_2 = 0.8\cos\theta$$

$$\theta = U[0,2\pi]$$

where $U[a,b]$ represents the uniform distribution in the range (a,b). The data set consisted of 100 data vectors of dimension two. Even though there is not a functional relationship between y_1 and y_2 (y_1 cannot be written as a unique function of y_2, and *visa versa*), NLPCA should be able to model the data with one nonlinear factor. Three methods were applied to this data set to determine the best one-factor representations: PCA, an autoassociative ANN with one hidden layer of sigmoidal nodes (ANN-1HL) and NLPCA.

Table 1. Results of One-Factor Representations for Example 1

Technique	Adjust. Param.	Error E	FPE	AIC
PCA	2	27.8	0.0708	−2.65
ANN, no mapping layers	7	26.4	0.0708	−2.65
NLPCA, no. mapping nodes				
2	19	10.5	0.0318	−3.45
3	27	1.35	0.00444	−5.42
4	35	0.348	0.00124	−6.70
6	51	0.336	0.00142	−6.57
8	67	0.307	0.00154	−6.50
10	83	0.302	0.00183	−6.36

Table 1 lists the number of parameters, the reconstruction error E, and the model selection criteria FPE and AIC for each method. The reconstruction errors for PCA and ANN-1HL are much higher than for any of the NLPCA networks. Since the training methods, node transfer functions, inputs, and outputs are the same, the difference between ANN-1HL and NLPCA is due solely to the mapping layers. This clearly shows that three hidden layers were required to achieve the desired results. To select the number of mapping nodes, the criterion of Eq. 10 was used in conjunction with the FPE and AIC model selection criteria. With the parameters of this problem, Eq. 10 gives $M_1 + M_2 \ll 49.5$. Table 1 shows that with more than four nodes in each mapping layer, the reconstruction error is approximately constant, and the FPE and AIC criteria indicate that four nodes in the mapping layers are optimal. The FPE and AIC also indicate that the NLPCA networks are preferred to the PCA and ANN-1HL representations.

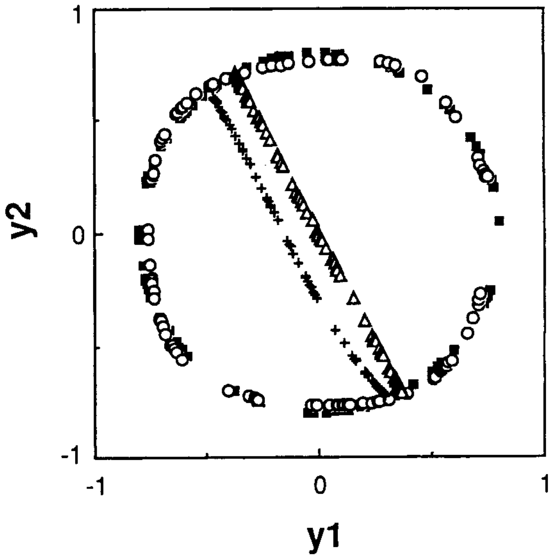


Figure 4. Reconstructed data from one factor, example 1.

Original data (■), reconstruction using four mapping and demapping nodes (o), reconstruction with no mapping nodes (+), reconstruction using PCA (△).

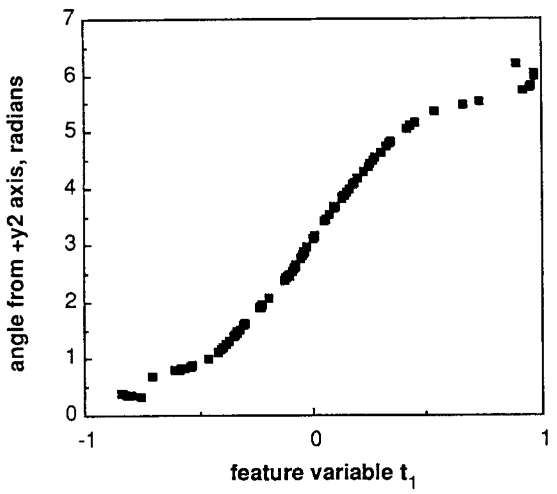


Figure 5. Feature variable derived by NLPCA, example 1, showing discovery of a feature analogous to underlying parameter θ.

Figure 4 shows the original training data and the reconstructions $\underline{Y}'$ using the three approaches. Because it is a linear technique, PCA produces a straight-line fit of the data, which is obviously inappropriate for the data. The ANN-1HL approach is only marginally better than PCA. Without mapping layers, this method can only produce features that are linear combinations of the inputs, compressed by the sigmoid function. The reconstruction by ANN-1HL is essentially linear but with a slight sigmoidal curve, again a poor fit. On the other hand, NLPCA (shown for $M_1 = M_2 = 4$) reconstructs the data from one nonlinear factor with great accuracy.

To successfully reconstruct the data from a single factor, NLPCA must have produced a feature analogous to the unmeasured parameter θ, since it is not possible to reconstruct the data from either input alone. The feature discovered by NLPCA is indeed related to the polar angle of the data point, specifically the angle measured counterclockwise from the positive y_2 axis. Figure 5 shows the relationship between the discovered feature variable (the output of the bottleneck layer of the trained network) and this angle. The relationship, although not linear, is monotonic, which suffices for reconstructing the input at the output layer. The network has in a sense "discovered" the concept of polar angle that is embedded in this data set.

Example 2

This example deals with data from a simulated batch reactor in which four simultaneous first-order reactions occur. The reaction network is shown in Figure 6. A is the raw material, R is the desired product, and S, T, and U are side products. Initially, only A is present. The reaction vessel is well-mixed and adiabatic. The reaction rate constants are:

$$k_1 = 10^7 e^{-5,000/T} [\text{h}^{-1}]$$

$$k_2 = \alpha \cdot 10^5 e^{-4,000/T} [\text{h}^{-1}]$$

$$k_3 = 10^{10} e^{-9,000/T} [\text{h}^{-1}]$$

$$k_4 = \alpha \cdot 10^8 e^{-7,000/T} [\text{h}^{-1}]$$

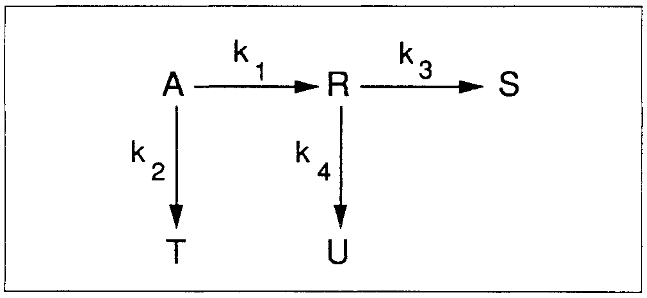
All reactions are thermally neutral except for the reaction $A \rightarrow R$, which is exothermic. The reaction parameters are such that complete reaction of A to R results in a 25° increase in reactor temperature. Each batch reacts for 0.5 hours, which brings R close to its optimal mole fraction of 0.75 under the given conditions. All five species concentrations are measured at 20 equally-distributed times during each batch. For the purposes of example, it is assumed that the initial temperature varies from batch to batch according to $T_0 = U[325,350]$. In addition, the reactions $A \rightarrow T$ and $R \rightarrow U$ are catalyzed by an impurity whose concentration varies randomly from batch to batch, modeled by $\alpha = U[0.5,1.5]$. All other factors in the operation remain fixed.

A data set containing data from 25 batches was prepared using this model. The data were scaled so that in each column of $\underline{Y}$, the mean was zero and the largest magnitude element was ± 0.9, the latter because the nodal transfer function requires infinitely large weights to reach ± 1. This scaling makes accurate reconstruction of all variables approximately equally important, similar to columnwise division by standard deviations sometimes used to prepare data sets for PCA.

The task is assumed to be representations of the data from each complete batch in a compact form, and discovery of the number of underlying parameters in the data. Since a total of 100 measurements are taken for each batch, the superficial dimension of the data is 100. The same variable measured at different times is represented as distinct inputs to the network so the features will represent the entire history of the batch. The features capture a temporal "window" of data and compress multivariable trajectories into a few features. (The same data could have been aligned as 500 examples of dimension five, with each data vector representing a single point in time. Rather than each batch appearing as a point in the feature space, a batch would be represented as a trajectory of 20 points in the feature space. This representation could be useful for dynamically "tracking" the progress of each batch.)

Table 2 shows the results for one-factor representations of this data. The reconstruction errors for PCA and ANN-1HL are both an order of magnitude higher than the optimal NLPCA networks. As in example 1, the FPE and AIC criteria indicate a preference for the NLPCA models. The constraint on the number of mapping nodes from Eq. 11 is $M_1 + M_2 \ll 25$. The FPE and AIC criteria indicate that six mapping nodes are optimal (approximately two data points for each adjustable parameter) among the architectures tested.

Results for the two-factor representations are similar. Again, PCA and ANN-1HL are virtually identical, while the error for



$$\begin{tikzcd} A \arrow[r, "k_1"] \arrow[d, "k_2"'] & R \arrow[r, "k_3"] \arrow[d, "k_4"'] & S \\ T & U & \end{tikzcd}$$

Figure 6. Reaction network for example 2.

Table 2. Results of One-Factor Representations for Example 2

Technique	Adjust. Param.	Error E	FPE	AIC
PCA	100	145.0	0.0314	−3.46
ANN, no mapping layers	301	144.9	0.0369	−3.30
NLPCA, no. mapping nodes				
2	509	120.6	0.0364	−3.31
4	917	52.1	0.0225	−3.83
6	1,325	15.9	0.0104	−4.69
8	1,733	15.8	0.0174	−4.37

NLPCA is an order of magnitude lower. These results are shown in Table 3.

To check the effectiveness of the one- and two-factor models, one batch run from the 25 was randomly selected for detailed study. Batch No. 9 had $T_0 = 334.6$ and $\alpha = 1.25$; its temporal history is shown in Figure 7. The scaled concentrations in this figure represent a normalization of the deviation of the concentration of species i from the concentration of i at time t in the average batch, $0.9\{C_{i9}(t) - \text{ave}_j[C_{ij}(t)]\}/\max_j|C_{ij}(t)|$, $1 \leq j \leq 25$. Figure 8 shows the errors as a function of time for four models: 1- and 2-factor PCA, and 1- and 2-factor NLPCA (using $M_1 = M_2 = 6$). The errors for 1-factor PCA are the same order of magnitude as the scaled concentrations themselves. For 2-PCA, there are still large errors, particularly for the most nonlinear trajectories. The errors for 1- and 2-factor NLPCA are much smaller and show little or no systematic error, particularly in the latter case. This is evidence that the intrinsic dimensionality of the data is two.

As in example 1, NLPCA must have discovered features analogous to the actual underlying parameters T_0 and α to describe the data with two features. To demonstrate this, the underlying parameter values for the 25 batches were plotted using different symbols, depending on the quadrant of the diagram in which they fell (Figure 9A). The batch data were identified according to these four classes, and a feature map (a plot of the output of the bottleneck layer) for the batches was prepared using the function $\underline{G}$ learned during NLPCA training. The feature representation of the batches is plotted in Figure 9B using the corresponding symbols from Figure 9A (the axes in Figure 9B are added for visual clarity and do not represent the feature space mapping of the axes in Figure 9A). It is evident from this diagram that the features discovered by NLPCA are related closely to the actual underlying parameters. The most significant difference is that each feature represents a combination of the actual parameters T_0 and α, accounting for the rotation in Figure 9B. The basic topology of the underlying parameter space, however, is clearly present in the feature space, notably the four quadrants of the actual parameter space map to four distinct quadrants of the feature space. On a more detailed level, there are certain groupings of points in Figure 9A that can be discerned clearly in the

Table 3. Results for Two-Factor Representations for Example 2

Technique	Adjust. Param.	Error E	FPE (×100)	AIC
PCA	200	25.7	0.604	−5.11
ANN, no mapping layers	502	25.6	0.770	−4.87
NLPCA, no. mapping nodes				
4	926	2.26	0.0985	−6.96
5	1,132	1.82	0.0965	−7.01
6	1,338	1.32	0.0869	−7.17
8	1,750	1.29	0.147	−6.86

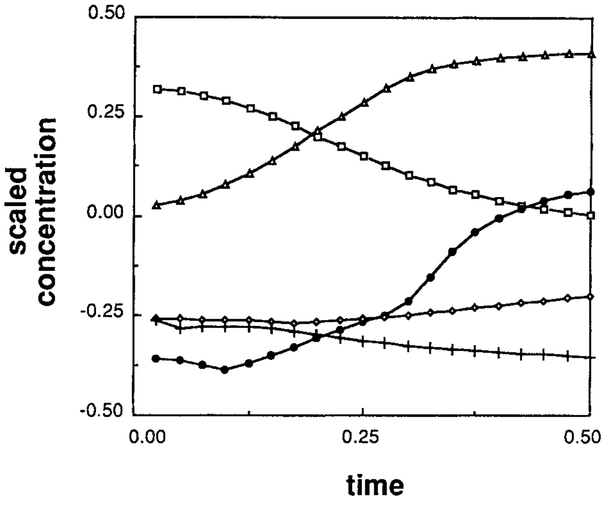


Figure 7. Time-concentration history of randomly-selected batch run.

$T_0 = 334.6$, $\alpha = 1.246$, Species A (—□—), T (—△—), R (—●—), U (—◇—), S (—+—)

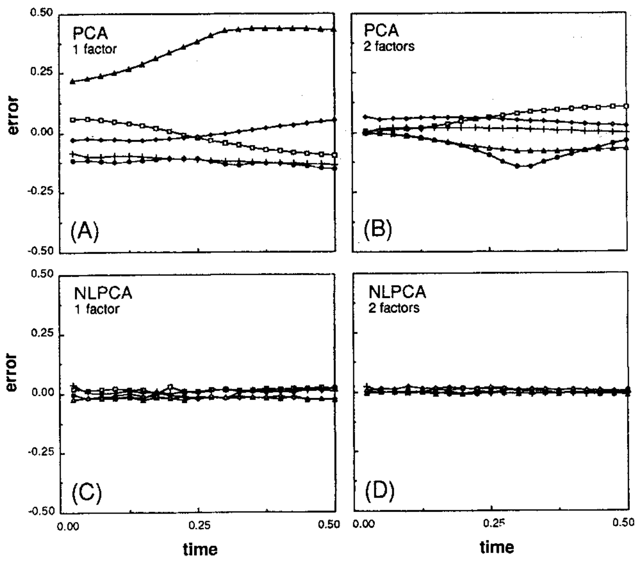


Figure 8. Errors in reproducing data for batch run in Figure 7 from feature space representations.

A. PCA method with one factor; B. PCA method, 2 factors; C. NLPCA, 1 factor; D. NLPCA, 2 factors. For symbols, see Figure 7.

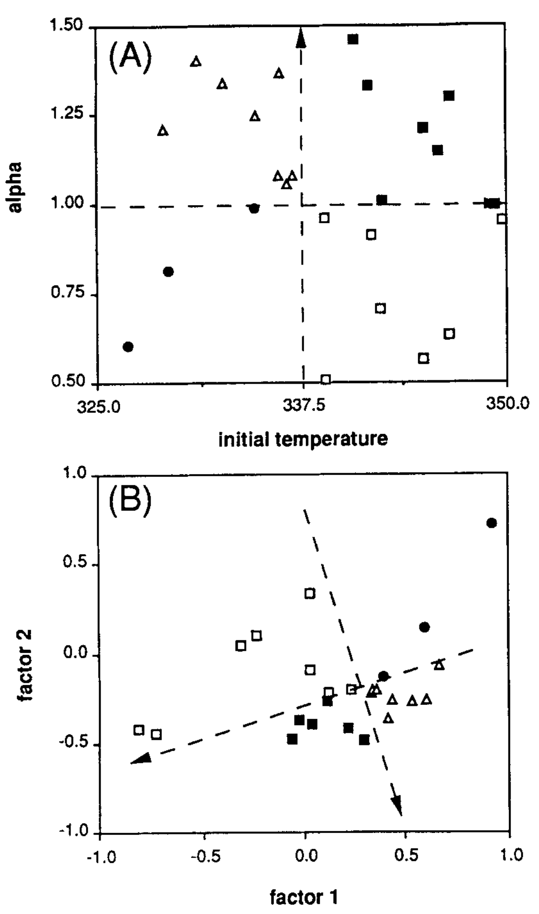


Figure 9. Underlying factors by NLPCA in example 2.

A. Original parameters; B. NLPCA feature space for corresponding batches. (■) High T_0, high α; (△) low T_0, high α; (●) low T_0, low α; (□) high T_0, low α

feature space representation. For example, in Figure 9A there are three batches at similar conditions represented by the three triangles near the center of the diagram. In Figure 9B, these batches appear grouped together and near the center of the diagram, indicated by the cross. The progression of three points represented as filled circles in the lower left of Figure 9A is also distinctive and appears in the feature space mapping. The feature map is a good representation of the actual parameter space. It was derived by NLPCA without any information on the character, number, or values of the underlying parameters, and without any knowledge of the process governing equations.

A feature map like Figure 9B can be valuable in developing process understanding. Given a measure of the success of the batches, the operator of the process can acquire a sense of where the "good" and the "bad" areas of the feature map are located. Each area of the map represents a different condition of operation. Learning to recognize "where the current batch is" relative to prior batches using the feature map may be easier than trying to make the same determination based on the raw data, because the trajectories display much more complicated features and the dimensionality is considerably higher. On the feature map, each batch is represented by a single point that is easily envisioned and related to prior batches.

Another use of the feature space representation is in forming correlations with other variables. Suppose we desire to train a new network to predict some output that is not immediately available in the form of an on-line measurement, for example, a quality variable. If the entire measurement vector is used as the network input, then the network will contain $(m+2)h+1$ weights, assuming the network contains one hidden layer with h nodes. Alternatively, a network trained to make the same prediction using the factors as inputs requires only $(f+2)h+1$ adjustable parameters. Since the factors contain the critical information from the inputs, the correlation model based on the features will be no more prone to error and will have higher significance than the model developed from all the inputs. In the current case, if the raw data are used to build the correlation model, $102h+1$ parameters would be required. Data on 25 batches are not sufficient to train the network. However, if the model is developed using the factors as inputs, then only $4h+1$ adjustable parameters are involved, greatly reducing the data required for training, and improving the significance of the resulting model.

Conclusion

The NLPCA method finds and eliminates nonlinear correlations in the data. Analogous to principal component analysis, this method can be used to reduce the dimensionality of data by removing redundant information. In theory, this method can remove any type of nonlinear correlation occurring in the data, since the basis functions utilized can model any bounded, continuous function to arbitrary accuracy. The results are limited only by the practicalities of computing functional approximations from limited data. The NLPCA methodology works by training a neural network containing a bottleneck layer to perform the identity mapping. The network architecture incorporates three hidden layers, which are necessary to achieve the general nonlinear fitting property.

Experiments with this methodology have shown it to be more effective than PCA in describing and reducing typical data. In the analysis of simulated batch reaction data, NLPCA outperformed PCA by a significant margin. The method was able to synthesize a topographically accurate map of the unmeasured parameters that determined the outcome of the batch, without any prior information of the existence or nature of the parameters.

Nonlinear PCA can be applied to the same problems as conventional PCA: data reduction and visualization, sensor validation, fault detection, quality control, principal component regression, etc. Because of NLPCA's ability to describe nonlinear data more efficiently than PCA, it should enhance the performance of these tasks.

Acknowledgment

The author wishes to thank Dr. Zvi Boger and Dr. Hugo Guterman for helpful discussions during the preparation of this article.

Notation

A	= chemical species
e	= average error, $E/2N$
E	= sum of squared errors, Eq. 9
$\underline{E}$	= matrix of residuals ($n \times m$)
f	= number of features or factors
$\underline{G}$	= mapping function
$\underline{H}$	= demapping function

n = number of data vectors
N = number of data points
N_w = number of weights and biases
m = number of input/output variables
M_1 = number of nodes in mapping layer
M_2 = number of nodes in demapping layer
$\underline{\underline{P}}$ = matrix of loadings ($m \times F$)
$\underline{p}_i$ = ith column of $\underline{\underline{P}}$
R = chemical species
$\Re^k$ = k-dimensional Euclidean space
S = chemical species
T = temperature (example 2)
T = chemical species
$\underline{\underline{T}}$ = matrix of scores ($n \times F$)
$\underline{T}$ = row of $\underline{\underline{T}}$
$\underline{t}_i$ = ith column of $\underline{\underline{T}}$
U = chemical species
w = network weights
$\underline{\underline{Y}}, \underline{\underline{Y}}_1$ = original data matrix ($n \times m$)
$\underline{\underline{Y}}_i$ = residual matrix after $(i-1)$ factors, $i > 1$ ($n \times m$)
$\underline{\underline{Y}}'$ = reconstructed data matrix ($n \times m$)
$\underline{Y}$ = single data vector

Greek letters

σ = sigmoid function
θ = nodal biases

Literature Cited

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

Baldi, P., and K. Hornik, "Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima," *Neural Networks*, **2**, 53 (1989).

Ballard, D. H., "Modular Learning in Neural Networks," *Proc. Conf. on AI* (AAAI-87), **1**, 279 (1987).

Chauvin, Y., "Toward a Connectionist Model of Symbolic Emergence," *Proc. Conf. on the Cognitive Sci. Soc.*, 580 (1989).

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

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

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

Fahlman, S. E., "Faster-Learning Variations on Back-Propagation: An Empirical Study," *Proc. of Connectionist Models Summer School*, D. Touretzky, G. Hinton, T. Sejnowski, eds., Morgan Kaufmann, San Mateo, CA (1989).

Földiák, P., "Adaptive Network for Optimal Linear Feature Extraction," Int. Joint Conf. on Neural Networks, Washington, DC, **I**, 401 (1989).

Fukunaga, K., and W. Koontz, "Application of Karhunen-Loeve Expansion to Feature Selection and Ordering," *IEEE Trans. Comput.*, **C-19**, 311 (1970).

Geladi, P., and B. R. Kowalski, "Partial Least-Squares Regression: a Tutorial," *Analytica Chimica Acta*, **185**, 1 (1986).

Haesloop, D., and B. R. Holt, "A Combined Linear/Non-Linear Neural Network for System Identification and Control," paper 96b, AIChE Meeting, Chicago (1990).

Hinton, G. E., "Learning Distributed Representations of Concepts," *Proc. Ann. Conf. of the Cognitive Sci. Soc.*, 1 (1986).

Hoskins, J. C., and D. M. Himmelblau, "Artificial Neural Network Models of Knowledge Representation in Chemical Engineering," *Comput. Chem. Eng.*, **12**, 881 (1988).

Joback, K. G., "A Unified Approach to Physical Property Estimation using Multivariate Statistical Techniques," MS Thesis, Mass. Inst. of Tech. (1984).

Karnin, E. D., "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks," *IEEE Trans. on Neural Networks*, **1**, 239 (1990).

Kruschke, J. K., "Creating Local and Distributed Bottlenecks in Hidden Layers of Back-Propagation Networks," *Proc. Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, eds., Morgan-Kaufman, San Mateo, CA (1989).

Leonard, J., and M. A. Kramer, "Improvement of the Backpropagation Algorithm for Training Neural Networks," *Comput. Chem. Eng.*, **14**, 337 (1990).

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

Mardia, K., J. Kent, and J. Bibby, *Multivariate Analysis*, Academic Press, London (1980).

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

Miikkulainen, R., and M. G. Dyer, "Encoding Input/Output Representations in Connectionist Cognitive Systems," *Proc. Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, eds., Morgan-Kaufman, San Mateo, CA (1989).

Mozer, M. C., and P. Smolensky, "Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment," *Advances in Neural Information Processing 1*, D. S. Tourezky, ed., Morgan-Kaufman (1989).

Niida, K., J. Tani, T. Hirobe, and I. Koshijima, "Application of Neural Network to Rule Extraction from Operation Data," Paper 6g, AIChE Meeting, San Franscisco (1989).

Oja, E., "A Simplified Neuron Model as a Principal Component Analyzer," *J. Math. Biology*, **15**, 267 (1982).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, **1**, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, MA (1986).

Sanger, T. D., "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, **2**, 459 (1989).

Siedlecki, W., K. Siedlecki, and J. Sklansky, "Mapping Techniques for Exploratory Pattern Analysis," *Pattern Recognition and Artificial Intelligence*, E. S. Gelsema and L. N. Kanal, eds., Elsevier Science Publ., Amsterdam (1988).

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

Venkatasubramanian, V., and K. Chan, "A Neural Network Methodology for Process Fault Diagnosis," *AIChE J.*, **35**, 1993 (1989).

Werbos, P. J., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," PhD Thesis, Applied Mathematics Dept., Harvard Univ. (1974).

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

Manuscript received Jan. 2, 1990, and revision received Dec. 27, 1990.