# A Simulation Study on Nonlinear Principal Component Analysis

by Brigitta Voss

Department of Statistics, University of Dortmund, D-44221 Dortmund, Germany

**Summary**

In statistical practice multicollinearity of predictor variables is rather the rule than the exception and appropriate models are needed to avoid instability of predictions. Feature extraction methods reflect the idea that latent variables not measurable directly are underlying the original data. They try to reduce the dimension of the data by constructing new independent variables which keep as much information as possible from the original measurements. A common feature extraction method is Principal Component Analysis (PCA), which in its classical form is restricted to linear relationships among predictor variables. This paper is concerned with nonlinear principal component analysis (NLPCA) as introduced by Kramer (1991), who modelled his approach with help of artificial neural networks. By means of first simulation studies data derived from semicircles and circles are investigated with respect to their ability to be described by nonlinear principal components among the predictors.

*Keywords*: feature extraction, nonlinear principal component analysis, artificial neural networks

# 1 Introduction

In many fields of applied statistics it is common practice to sample many predictor variables while hoping that they may be useful for describing and investigating a poorly known processes and for predicting one or more response variables, e.g. in technical areas. Unless these predictors are collected according to an experimental design, they tend to be correlated. The presence of these multicollinearities among the predictor variables can be caused by predictors being measurements of underlying latent variables, that are not measurable directly. Therefore it might be a promising attempt to extract a new set of variables, so-called feature variables or scores. These feature variables are functions of the observed measurements, extracting most of the information needed for describing the process and can be used for prediction afterwards. The feature variables contain nearly the same information but within smaller dimension.

Thus we may assume that the new scores are related to the observed data matrix $\mathbf{x} \in \mathbb{R}^{p\times n}$ as follows:

$$\mathbf{x} = \mathbf{f}(s) + \epsilon = \begin{pmatrix} f_1(s) + \epsilon_1 \\ \vdots \\ f_p(s) + \epsilon_p \end{pmatrix}, \tag{1.1}$$

where $\mathbf{f}$ describes an $r$-dimensional surface in $\mathbb{R}^p$, $s$ is the score and the vector $\epsilon$ describes noise (Malthouse, 1995). The feature extracting problem is to find $\mathbf{f}$ and $s$.

The problem of feature extraction is closely related to the problem of dimensionality reduction. The superficial dimension of the observed data is much greater then its intrinsic dimension, the number of independent underlying variables, describing the significant variables in the observations.

Within the class of linear feature extraction methods, principal component analysis (PCA) gives the optimal information preserving transformation (Fukunaga & Koontz, 1970). In PCA feature variables are linear combinations of the original variables.

As for PCA, most of the methods for feature extraction have been developed for linear relationships between the predictor variables, but in many applications they are connected nonlinearly. Therefore, too many linear feature variables are needed to approximate these nonlinear relationships by using PCA.

There have been several attempts for generalizing PCA, but this article is focusing only on the nonlinear principal component analysis as introduced by Kramer (1991). After this introduction, a review on linear PCA is given. Section 3.1 gives a short description of the types of neural networks used in this context here and Section 3.2 describes the relationship between artificial neural networks and PCA. Section 4 introduces the nonlinear principal component analysis (NLPCA) developed by Kramer. Simulation studies investigating the performance of NLPCA for different kinds of nonlinear relationships are described in Section 5.

## 2 Linear Principal Component Analysis

The idea of principal component analysis (PCA, Johnson & Wichern, 1992; Mardia, Kent & Bibby, 1979) is to find so-called scores, describing most of the variability in the data. So PCA is concerned with explaining the variance-covariance structure through a few linear combinations of the original variables. The general objective is data reduction to improve interpretation, and the scores are often used for explaining and predicting dependent variables (principal component regression, see Schmidli, 1995).

In the following we consider a vector $\mathbf{X}$ of $p$ random variables $X_1, \ldots, X_p$ and the data matrix $\mathbf{x} \,\epsilon\, I\!R^{p \times n}$ with $n$ observations (columns) and $p$ variables (rows), centered with respect to the sample mean vector. Denote the $i$th column vector of $\mathbf{x}$ by $\mathbf{x_i} \,\epsilon\, I\!R^p$.

Algebraically, principal components are linear combinations of the $p$ random variables; its scores are built by a linear combination of their observations. The first principal component is the normalized linear combination of the $p$ variables $X_1, \ldots, X_p$ with the largest variance:

$$\mathbf{u_1} : \max(\hat{Var}(\mathbf{X}'\mathbf{u_1})) \text{ with } \mathbf{u_1'u_1} = 1. \tag{2.1}$$

The vector $\mathbf{s_1}$ containing the $n$ so-called scores for the first principal component is then given as the corresponding linear combination of the observed data: $\mathbf{s_1} = \mathbf{x}'\mathbf{u_1}$. The second principal component is chosen to have the highest variance among all directions orthogonal to the first principal:

$$\mathbf{u_2} : \max_{\mathbf{u_2'u_2}=1} Var(\mathbf{X}'\mathbf{u_2}) \text{ with } Cov(\mathbf{X}'\mathbf{u_1}, \mathbf{X}'\mathbf{u_2}) = 0. \tag{2.2}$$

and again the score vector is given by $\mathbf{s_2} = \mathbf{x}'\mathbf{u_2}$. The $i$th principal component is then given by:

$$\mathbf{u_i} : \max_{\mathbf{u_i'u_i}=1} \hat{Var}(\mathbf{X}'\mathbf{u_i}) \text{ with } \hat{Cov}(\mathbf{X}'\mathbf{u_i}, \mathbf{X}'\mathbf{u_j}) = 0 \ \forall j < i. \tag{2.3}$$

The principal components are obtained by computing the eigen decomposition of the sample covariance matrix $\mathbf{C\hat{o}v}$:

$$\mathbf{C\hat{o}v} = \sum_i \hat{\lambda}_i \mathbf{e_i e_i'} \tag{2.4}$$

with $\hat{\lambda}_i$ the $i$th estimated eigenvalue and $\mathbf{u_i} = \mathbf{e_i}$ the normalized eigenvector belonging to $\hat{\lambda}_i$, which is also called direction vector, because it gives the direction of the $i$th highest variation in the data and thus forms an element of a new coordinate system. The $i$th estimated eigenvalue $\hat{\lambda}_i$, gives the proportion of total variance in the data explained by the $i$th principal component. If most of the total variance can be attributed to the first $r$ components, then these components can replace the original $p$ variables without much loss of information (see Johnson & Wichern, 1992).

For a geometrical approach to PCA, it might be convenient to think of $\mathbf{x}$ as a cloud of $n$ points in $p$-dimensional space. PCA reduces the data to its intrinsic dimension by fitting an $r$-dimensional plane through the middle of the points, so that the sum of the distances between the points $\mathbf{x_i}$ and their projections $\tilde{\mathbf{x}}_\mathbf{i}$ onto the plane is minimized. This hyperplane is found by using the first $r$ eigenvectors $\mathbf{U_r} = (\mathbf{u_1}, \ldots, \mathbf{u_r})$ (where the columns $\mathbf{u_j}$, $j = 1, \ldots, r$, of $\mathbf{U_r}$ denote the $r$ unit-length eigenvectors), forming a basis for $I\!R^r$, because the matrix of eigenvectors minimizes the following quantity among all $p \times r$ matrices $\mathcal{M}$ (Mardia et al., Section 8.2.3d):

$$\mathbf{U_r} = \min_{\mathcal{M}} \|\mathbf{x} - proj_{\mathcal{M}}\mathbf{x}\|^2, \tag{2.5}$$

where $proj_{\mathcal{M}}\mathbf{x}$ denotes the projection of $\mathbf{x}$ onto a subspace spanned by matrix $\mathcal{M}$. This means that PCA approximates $\mathbf{x}$ by projecting it onto an $r$-dimensional subspace. For the first principal component, the direction vector $\mathbf{u_1}$ represents the direction with maximum variability in the data, that minimizes the sum of squared distances between $\mathbf{x_i}$ and their projected points $\tilde{\mathbf{x}}_{\mathbf{i},\mathbf{u_1}}$ on $\mathbf{u_1}$. Here we assume an orthogonal projection of the $i$th observation point $\mathbf{x_i}$ onto the direction vector $\mathbf{u_1}$:

$$\tilde{\mathbf{x}}_{\mathbf{i},\mathbf{u_1}} = proj_{\mathbf{u_1}}(\mathbf{x_i}) = (\mathbf{x_i'u_1}/\mathbf{u_1'u_1})\mathbf{u_1}, \tag{2.6}$$

where the product $(\mathbf{x_i'u_1}/\mathbf{u_1'u_1})$ denotes for each point $\mathbf{x_i}$ the length from the projection point to the origin and $\mathbf{u_1}$ their direction. As we assume normalized direction vectors the orthogonal projection of Equation (2.6) is given by $(\mathbf{x_i'u_1})\mathbf{u_1}$. Thus the geometrical interpretation of the score values $s_{1i} = \mathbf{x_i'u_1}$ is the length of this projection. The orthogonal projection of $\mathbf{x}$ is then given by

$$\tilde{\mathbf{x}}_{\mathbf{u_1}} = proj_{\mathbf{u_1}}(\mathbf{x}) = ((\mathbf{x_1'u_1})\mathbf{u_1}, \ldots, (\mathbf{x_n'u_1})\mathbf{u_1}) = \mathbf{u_1u_1'x}. \tag{2.7}$$

Points on the same hyperplane orthogonal to the direction vector will therefore have the same score value. This seems intuitively appealing, as they contain the same information about the variation of the data in direction of $\mathbf{u_1}$. With this, the $r$-dimensional coordinates of the points $\mathbf{x_i}$ relative to the eigenbasis $\mathbf{U_r}$ are given by the $r$ scores $\mathbf{s_{(i)}} = \mathbf{x_i'U_r}$ .

# 3 Artificial Neural Networks

This section gives a short motivation for the use of artificial neural networks in this context here. First a more general review of feedforward neural networks will be given. Afterwards Section 3.2 shows an approach to perform principal component analysis with help of neural networks, e.g. how neural networks and PCA are related.

## 3.1 Feedforward Artificial Neural Networks

In the following, three-layer neural networks (NN) with feedforward connection are described. These kinds of NN are frequently used for approximating functional relations.

Such neural networks describe a class of models with functions

$$Y = \sum_{h=1}^{H} \alpha_h \mathcal{A}(\sum_{j=1}^{p} X_j w_{jh}) + \epsilon = f(\mathbf{X}; \theta) + \epsilon, \tag{3.1}$$

which can be graphically displayed as seen in Figure 1, using $H$ so-called

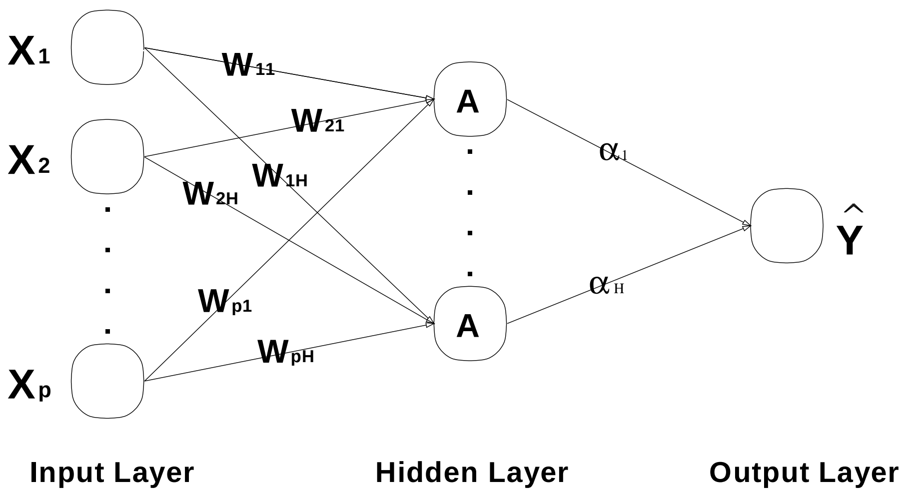


Figure 1: *Three-layer neural network with $H$ hidden nodes.*

hidden nodes, where $\mathcal{A}$ describes the so-called activation function.

A feedforward NN is restricted to all signals going in one direction, from input cells to output cells. As seen in Figure 1, networks can be organized hierarchically into layers of neurons. The connection between two cells has a numerical value, called weight, representing the influence of the input cell on the output cell. The input signals are combined linearly with respect to various weights to obtain input signals for the second layer. These input signals are then passed through an activation function $\mathcal{A}$, to yield output signals of the cells on the second layer.

To approximate linear functions the identity function

$$\mathcal{A}(x) = l(x) = x \tag{3.2}$$

is often chosen, while the ability of neural networks to fit arbitrary nonlinear functions depends on the presence of hidden layers with nonlinear nodes. An especially popular choice of activation function for nonlinear problems is the sigmoidal function

$$\mathcal{A}(x) = \sigma(x) = \frac{1}{1 + exp(-x)}. \tag{3.3}$$

These are the two activation functions used for our problems here. The parameters of equation (3.1), e.g. the weight values of the NN are determined by minimizing the following least squares objective function:

$$\min_{\alpha_h, w_{jh}, h=1,\dots,H, j=1,\dots,p} \sum_{i=1}^{n} \left[ \ \|\mathbf{y_i} - (\sum_{h=1}^{H} \alpha_h \mathcal{A}(\sum_{j=1}^{p} x_{ij} w_{jh}))\|^2 \right]. \qquad (3.4)$$

While training the network, the weights are successively modified, according to several possible training algorithms. Cybenco (1989) showed that with help of 3-layers-neural-networks with sigmoidal activation function every continuous function can be approximated to an arbitrary degree of precision. The approximation further improves with increasing number of hidden nodes.

## 3.2 Principal Component Analysis and Neural Networks

The use of feedforward neural networks to extract principal components are described by Baldi and Hornik (1989) using a network structure shown in Figure 2. This three-layer neural network has $p$ nodes in both, input and output layers, and $r < p$ nodes in the hidden layer for estimating $r$ principal components. Here the identity function is used to force the network to approximate a linear function.

Figure 2 represents a special case with only one hidden node, e.g. only one principal component is estimated. Networks trained to reproduce their inputs in the output layer are called autoassociative neural networks, e.g. they perform the so-called identity mapping. These kinds of networks are typically used for tasks involving pattern completion (e.g. Ballard, 1987).

The hidden layer in these autoassociative NN is called the bottleneck layer, because it causes the NN to summarize the information in the input variables

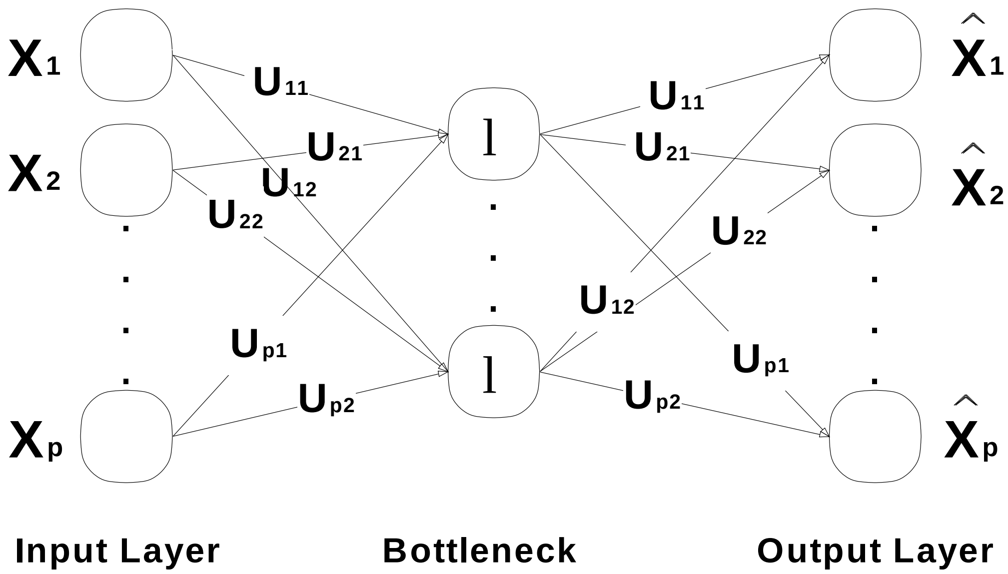


Figure 2: *Neural network for principal component analysis.*

to $r < p$ dimensions, e.g. to $r$ principal components. Since there are fewer nodes in the hidden layer than in the output, the bottleneck nodes must represent the most important information of the input (Sanger, 1989). If the network training leads to an acceptable solution, e.g. estimates the input right, a good representation of the input data must exist in the bottleneck layer. This implied data compression caused by the network bottleneck may force hidden units to represent significant features in data.

As seen in Figure 2 the weights leading from the $j$th input node to the $k$th hidden node are the same as the ones going from the $k$th hidden node to the $j$th output node. So there is only one weight matrix $\mathbf{U}$ with $p$ rows and $r$ columns, containing the weights of the networks. The $k$th column $\mathbf{u_k}$ $k = 1, \ldots, r$ contains the weights for the signals leading to the $k$th node in the hidden layer. The weights for the signals going to the $j$th output node are given by the $j$th row of $\mathbf{U}$, $j = 1, \ldots, r$. With this, the outputs of the network are then given by $\hat{\mathbf{x}} = \mathbf{U}\mathbf{U}'\mathbf{x}$.

Note that the architecture of this NN estimates the PCA solution because it minimizes the same objective functions, as given in Equations (2.5) and (2.6):

$$\min_{\mathbf{U}} = \sum_{i=1}^{n} \sum_{j=1}^{p} \|x_{ij} - (\mathbf{x_i'U})\mathbf{u'_{(j)}}\|^2, \tag{3.5}$$

where $\mathbf{u_{(j)}}$ denotes the $j$th row of matrix $\mathbf{U}$. The $r$ weight vectors $\mathbf{u_k}$ between the input and bottleneck layer, given as columns of matrix $\mathbf{U}$, span the same subspace as the first $r$ eigenvectors in Equation (2.4). After training, the NN weights can be orthonormalized without changing the value of the objective function (Baldi & Hornik (1989)). Baldi and Hornik also proved that this linear network has a unique minimum, e.g. there is just one matrix $\mathbf{U}$ solving Equation (3.5).

# 4 Nonlinear Generalizations

The linear principal component analysis (PCA) assumes the relationship between the observed variables and the feature variables to be linear, e.g. a change in the observed variable is associated with a proportional increase in the feature variable. It is easy to assume situations, where a nonlinear relationship exists. When looking at nonlinear data, it would be useful to generalize the principal components to nonlinear curves and surfaces, describing the structure of the data in fewer dimensions, than by using linear combinations as it is done in PCA.

The nonlinear principal component analysis (NLPCA) as introduced by Kramer (1991), extends PCA by relaxing the assumption that $\mathbf{u}$ is linear. Section 4.2 summarizes this method. But first, a short review about differential geometry used in the following will be given in Section 4.1. For an extensive description see Bronstein et al. (1979), Hastie (1984) or Thorpe (1979).

## 4.1 Preliminary Remarks on Nonlinear Curves

Kramers approach to generalize linear principal component analysis is to build scores by projecting observation points $\mathbf{x_i}$ onto a curve or a surface instead of a vector $\mathbf{u}$. Here an $r$-dimensional nonlinear surface in $p$-dimensional space refers to a vector $\mathbf{f}(\mathbf{s})$ of $p$ nonlinear smooth functions of $r$ variables:

$$\mathbf{f} : \mathbf{A} \subset I\!R^r \to \mathbf{B} \subset I\!R^p \qquad \text{with} \qquad \mathbf{f}(\mathbf{s}) = \begin{pmatrix} f_1(s_1, \ldots, s_r) \\ \vdots \\ f_p(s_1, \ldots, s_r) \end{pmatrix}. \tag{4.1}$$

The parameter vector $\mathbf{s}$ describes the location of point $\mathbf{x_i}$ relative to the parameterization of surface $\mathbf{f}$. When $s$ is unidimensional, surface $\mathbf{f}$ is called a curve.

Sometimes it might be convenient to parameterize a curve by its arc length, that means each point on the curve can be described by its length along the curve starting at the origin, as it is done in linear PCA too. Using calculus, the arc length of curve $\mathbf{f}$ from $s_0$ to $s_1$ is given by:

$$l = \int_{s_0}^{s_1} \sqrt{\sum_{j=1}^{p} \left( \left. \frac{\partial f_j}{\partial s} \right|_{s=z} \right)^2} dz. \tag{4.2}$$

As the definition of a curve is not unique there are many different functions $\mathbf{f}$ that define the same curve, but with different parameterizations. Hence, an additional property for the uniqueness of this parameterization is needed. Therefore a curve is parameterized by arc length iff it fulfills the unit-speed property:

$$\sum_{j=1}^{p} \left( \frac{\partial f_j}{\partial s} \right)^2 = 1. \tag{4.3}$$

This property implies $\mathbf{f}$ to be a vector of smooth functions, because the slope of each function must be between -1 and 1. Unit-speed-curves define their length between the origin and point $s$ by the value $s$ itself. From differential geometry it is known that every smooth curve can be parameterized by arc length (Malthouse, 1995).

## 4.2 Nonlinear Principal Component Analysis

Kramer (1991) proposed two different types of nonlinear principal component analysis (NLPCA): sequential NLPCA and simultaneous NLPCA. While simultaneous NLPCA is looking directly for an $r$-dimensional surface to summarize the information in the $p$-dimensional data set, sequential NLPCA is more adapted to linear PCA by estimating iteratively $r$ one-dimensional curves. Therefore, the geometrical idea and its analytic solution of sequential NLPCA is described first. Afterwards a short remark on the generalization to simultaneous NLPCA is given.

### 4.2.1 Sequential Nonlinear Principal Component Analysis

The sequential NLPCA (hereafter simply refered to as NLPCA) generalizes the idea of the first principal component to a unit-speed curve, i.e. a curve $\mathbf{f_1}$ through the data points minimizing the sum of squared distances between the observed data points and the curve is sought. The NLPCA fits a composition of two functions, $s_{f_1} : \mathbb{R}^p \to \mathbb{R}$, the so-called projection-index, and $\mathbf{f_1} : \mathbb{R} \to \mathbb{R}^p$ the curve through the data points. Ideally, the projection-index maps each observation point to a point on the curve that is closest to it. In difference to linear principal component analysis no orthogonal projection is assumed. The definition for the projection-index $s_f$ for a given curve $\mathbf{f}$ used in our context is given by (Hastie, 1984):

$$s_f(\mathbf{x}) = \sup_s \{s : \|\mathbf{x} - \mathbf{f}(s)\| = \inf_\mu \|\mathbf{x} - \mathbf{f}(\mu)\|\}. \tag{4.4}$$

This definition means that the projection-index $s_f$ evaluated at $\mathbf{x_i}$ denotes the score value $s$ for which curve $\mathbf{f}(s)$ is closest to $\mathbf{x_i}$. If there are several such values for $s$, so called ambiguous points, by definition the largest one is selected (Kramer, 1991).

With this definition, it means geometrically for the NLPCA, that each data point $\mathbf{x_i}$ is projected on a point $\mathbf{\tilde{x}_{i,f_1}}$ on the curve $\mathbf{f_1}$, that is next to $\mathbf{x_i}$. The score $s_{i,f_1}$ of $\mathbf{x_i}$ is then given by the arc length between the projection point $\mathbf{\tilde{x}_{i,f_1}}$ and the origin, as it is also done in linear PCA described above. So for the first curve $\mathbf{f_1}$ each projection point can be described by its one-dimensional coordinate $s_{i,f_1}$ or by its $p$-dimensional coordinate $\mathbf{f_1}(s_{i,f_1}) = \mathbf{\tilde{x}_{i,f_1}}$. With finding a curve $\mathbf{f_1}$ passing through the middle of the data points

$$\min_{\mathbf{f_1}, s_{f_1}} \sum_{i=1}^{n} \left[ \; \|\mathbf{x_i} - \mathbf{f_1}(s_{f_1}(\mathbf{x_i}))\|^2 \right], \tag{4.5}$$

the composition $\mathbf{f_1}(s_{f_1}) = \mathbf{\tilde{x}_{i,f_1}}$ smoothes the data.

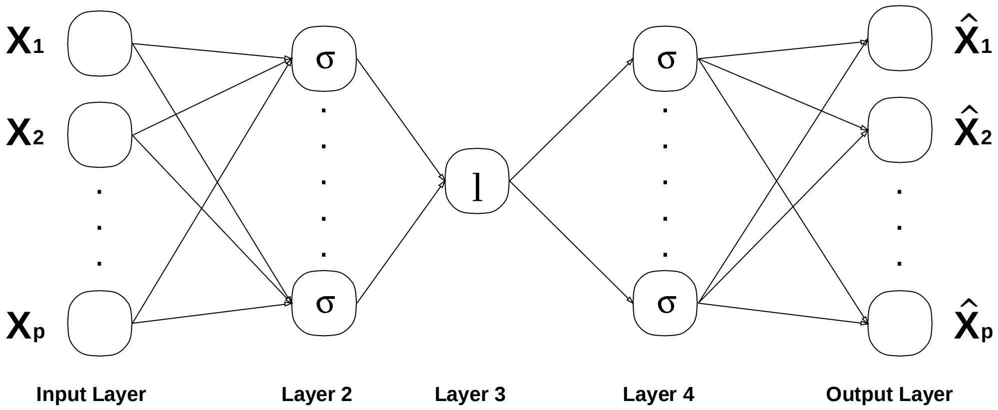


Figure 3: *Neural network for principal component analysis, where $\sigma$ denotes a sigmoidal activation function and $l$ the identity function.*

The functions $s_{f_1}$ and $\mathbf{f_1}$ are modeled by two connected three-layer neural networks. Therefore a five-layer-neural network (Figure 3) is used to model the composition of functions $\mathbf{f_1}(s_{f_1}(\mathbf{x_i}))$. The NLPCA network has $p$ nodes in the input layer, one node in the third layer, the so-called bottleneck layer, and again $p$ nodes in the output layer. The subnetwork consisting of layers 1 to 3 models function $s_{f_1}$, while layers 3 to 5 model surface $\mathbf{f_1}$. The nodes in layers 2 and 4 must have nonlinear activation functions (see Equation 3.3) to represent arbitrary smooth functions, while the layers 3 and 5 have linear activation functions (here the identity function is chosen, see Equation 3.2). The data reduction takes place because the $p$-dimensional input is forced to pass through the one dimensional bottleneck before reproducing the inputs. The first three-layer network ($s_{f_1}$) reduces the $p$-dimensional input data to the one-dimensional scores, while the second three-layer network $\mathbf{f_1}$ gives estimates of the input vectors from the scores.

The five-layer neural network is trained to reproduce its inputs under the following objective function (Malthouse, 1995):

$$\min_{f_1, s_{f_1}} \sum_{i=1}^{n} \left[ \; \|\mathbf{x_i} - \mathbf{f_1}(s_{f_1}(\mathbf{x_i}))\|^2 + \left( \sum_{j=1}^{p} \left( \frac{\partial f_{1_j}(\mathbf{x_i})}{\partial s} \right)^2 - 1 \right) \right]. \tag{4.6}$$

A penalty term is added to Equation (4.5), which forces the network to produce curves with unit-speed. Once the network has been trained, the bottleneck node activation value gives the score.

After estimating $\mathbf{f_1}$ and $s_{f_1}$, residuals of the data matrix $\mathbf{x} = \mathbf{e_o}$ are computed: $\mathbf{e_{1i}} = \mathbf{e_{0,i}} - \mathbf{f_1}(s_{f_1}(\mathbf{e_{0,i}}))$. This means that for the next step, estimating $\mathbf{f_2}$ and $s_{f_2}$, the data matrix $\mathbf{x}$ is replaced by its residual matrix $\mathbf{e_1}$ and this sequential procedure is repeated $r$ times, until the residuals are sufficiently small.

After finding $r$ curves describing the structure of the data, each data point can be described by its $r$-dimensional coordinates (Malthouse, 1995):

$$(s_{f_1}(\mathbf{e_{0,i}}), \ldots, s_{f_r}(\mathbf{e_{r-1,i}})). \tag{4.7}$$

Malthouse states "that one major problem of this sequential procedure is, that it is not clear, what removing a nonlinear direction from a matrix means" (see Malthouse, 1995). Since, the functions $\mathbf{f_1}, \ldots, \mathbf{f_r}$ are not orthogonal to each other, it might be more difficult, sometimes impossible, to estimate surfaces stepwise by using residual matrices, instead of estimating the $r$-dimensional surface at once.

### 4.2.2 Simultaneous Nonlinear Principal Component Analysis

In difference to sequential NLPCA the simultaneous NLPCA estimates an $r$-dimensional surface directly to summarize the information in the data set. The projection-index $\mathbf{s_f} : \mathbb{R}^p \rightarrow \mathbb{R}^r$ maps each observation on a surface $\mathbf{f} : \mathbb{R}^r \rightarrow \mathbb{R}^p$ and by this gives to each $\mathbf{x_i}$ the $r$-dimensional coordinates of the projected point $\tilde{\mathbf{x}}_\mathbf{i}$ on $\mathbf{f_i}$ in one step. These coordinates are then used as scores. This is realized by using $r$ nodes in the third layer of the network instead of one. As with the PCA network, data compression takes place because the $p$ inputs pass through the $r < p$ dimensional bottleneck layer before reproducing the inputs.

# 5 Simulation Studies

In most of the articles in this field of research, neural networks are used as a tool for solving minimization problems and estimating functions. Most of the time this is done by applying neural networks as a black box. The simulations described in this section are an attempt to treat artificial neural networks as a system of nonlinear equations, and it will be tried to solve them with SAS.

For a good visualization of the results only two variables are assumed. This means, that only one nonlinear principal component is to be estimated, so that both types of NLPCA, sequential and simultaneous, are equal.

According to Equation (3.1), we can view this network as a system of non-linear equations for estimating the functions $\mathbf{f}$ and $s_f$. The projection-index $s_f$ is given by the layers 1 to 3 of the network

$$\hat{s}_f(\mathbf{x_i}) = \sum_{a=1}^{A} v_a \frac{1}{1 + exp(-\mathbf{x_i'u_a})} \quad , \tag{5.1}$$

assuming $a = 1, \ldots, A$ nodes in the 2nd layer. The $j = 1, \ldots, p$ input nodes are connected by the weights $u_{ja}$ to the nodes of the second layer, where the $v_a$ are weighting the signals between the 2nd layer and the bottleneck, layer 3.

The curve $\mathbf{f}$, describing the first principal component, is given by the network layers 3 to 5

$$\hat{\mathbf{f}}(\hat{s}) = \begin{cases} \hat{f}_1(s) = \sum_{c=1}^{C} z_{c1} \frac{1}{1+exp(-w_c \hat{s})} \\ \hat{f}_2(\hat{s}) = \sum_{c=1}^{C} z_{c2} \frac{1}{1+exp(-w_c \hat{s})} \end{cases} \quad , \tag{5.2}$$

with $c = 1, \ldots, C$ nodes in the 4th layer. The weights leading from the 3rd to the 4th layer are denoted by $w_c$ and $z_{cj}$ stand for the weights between the 4th and the output layer. Thus, the $p = 2$ coordinates of the projected points $\tilde{\mathbf{x}}_\mathbf{i}$ are estimated using the following functions:

$$\begin{aligned}\hat{x}_1 = \hat{f}_1(\hat{s}_f(\mathbf{x_i})) = \sum_{c=1}^{C} z_{c1} \frac{1}{1+exp(w_c(\sum_{a=1}^{A} v_a \frac{1}{1+exp(-\mathbf{x_i'u_a})}))} \\ \hat{x}_2 = \hat{f}_2(\hat{s}_f(\mathbf{x_i})) = \sum_{c=1}^{C} z_{c2} \frac{1}{1+exp(w_c(\sum_{a=1}^{A} v_a \frac{1}{1+exp(-\mathbf{x_i'u_a})}))}\end{aligned} . \tag{5.3}$$

To fulfill the unit-speed-property (Equation 4.3), we wish to add an additional equation to this system:

$$\sum_{j=1}^{2} \left(\frac{\partial f_j}{\partial s}\right)^2 = \sum_{j=1}^{2}(\sum_{c=1}^{C} z_{cj} \frac{w_c exp(-w_c s)}{(1 + exp(-w_c s))^2})^2 = 1. \tag{5.4}$$

But for a network with more than one hidden node in the 2nd and 4th layer, this constraint exceeds the capability of SAS. So for first simulations we generate data from unit-speed curves and ignore the restriction.

The simulations described here are done by using the nonlinear equation system given in Equations (5.2) and (5.3) solved by SAS 6.12 applying Proc Model using the Newton-Marquart-method for estimating the parameter with 100 iterations per run. First simulations studies indicate that using 100 iterations is a good first value.

Because solving this equation system means to estimate the functions $\mathbf{f}$ and $s_f$, the assessment of the network training is judged by the values of goodness of fit $R^2$ for $x_1$ and $x_2$ (given in Table 1).

## 5.1 Estimation of Semicircles

As a first step for our simulations, we used a semicircle, with standard normal random errors $e_i$ added to both coordinates, $x_{1i} = cos(i) + e_i/10$ and $x_{2i} = sin(i) - 0.637 + e_i/10$ with $i \,\epsilon\, [0, \pi]$ in steps by 0.001; so 3142 data points are simulated. By these simulations we found, that for a successful

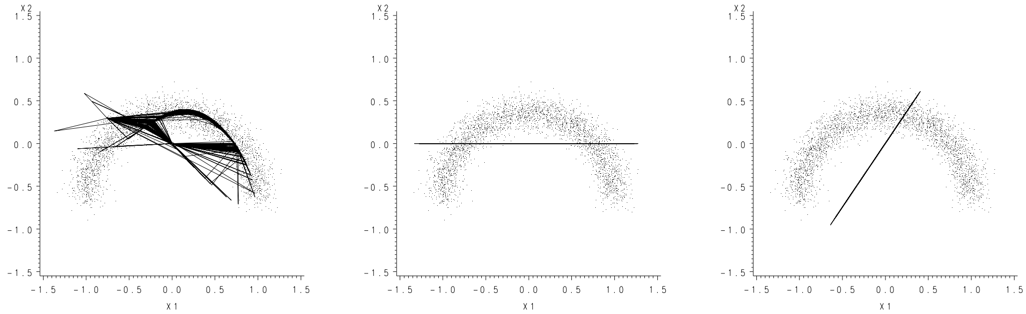


Figure 4: Estimated semicircle with 2 nodes in the 2nd and 4th layer each.

training of the network, i.e. solving this equation system, finding good starting values is key. This is especially true when using small neural networks, with only a few hidden nodes. For different starting values and a network with $A = 2$ nodes in the 2nd layer and $C = 2$ nodes in the 4th layer we got unsatisfactory estimates of the functions according to Figure 4.

Since we know the underlying process, given by the equation of the semicircle, we split this estimation problem by training the two subnetworks, layer 1 to 3 estimating the scores and layer 3 to 5 estimating $\hat{\mathbf{x}}_\mathbf{i}$ out of the scores separately. Afterwards the five-layer-network was trained again using the parameters estimated in this two subnetworks as starting values. This procedure was quite capable to learn the semicircle.

Estimating the semicircle without prior knowledge about the scores we needed

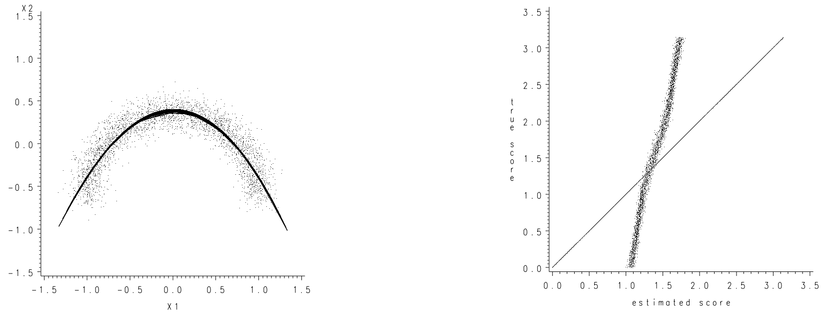


Figure 5: Estimated semicircle with 3 nodes in the 2nd and 4th layer (left) and its estimated scores compared to the underlying score (right).

at least $A=3$ and $C=3$ nodes (Figure 5). In this case the larger number of nodes seems to prevent the network from converging to local minima that easily, but results in an increased number of parameters to be estimated. For this network, increasing the number of nodes by one in layer 2 or 4 requires additional estimation of $p+1$ parameters. For the simulations described here the number of nodes in the 2nd and 4th layer are increased equally; so additional $2\,(p+1)$ parameters are needed to be estimated. Because nonlinear PCA is proposed as a method for data sets with many variables, any extension of the network resulting in a markedly increased number of parameters to be estimated may be critical in practical applications.

For most of the estimated semicircles the values of goodness of fit $R^2$ are for $x_1$ next to 1 (for Figure 5 0.99), but for $x_2$ next to 0.8 (here 0.83, see Table 1).

For the estimation of the semicircle, given in Figure 5 (left), we looked closer at the estimated scores, given in the bottleneck layer and compared them to the underlying "scores" of the semicircle. It appeared, that the range of the estimated scores is much smaller than that of the original score (Figure 5, right). Note, that the diagonal line indicates equal range.

Table 1: *Goodness of fit measures for the trained networks*

| | $R^2$ for $\hat{x}_1$ | $R^2$ for $\hat{x}_2$ |
|---|---|---|
| semicircle trained with 3 nodes (Figure 5) | 0.99 | 0.83 |
| circle trained with 6 nodes (Figure 7) | 0.98 | 0.98 |
| circle trained with 7 nodes (Figure 8) | 0.95 | 0.91 |
| parable trained with 6 nodes (Figure 10) | 0.89 | 0.99 |

We thought, that it might be easier for a network, and therefore fewer nodes are necessary, to estimate only positive or negative values out of the score. Assuming a neural network with only one hidden node in the last hidden

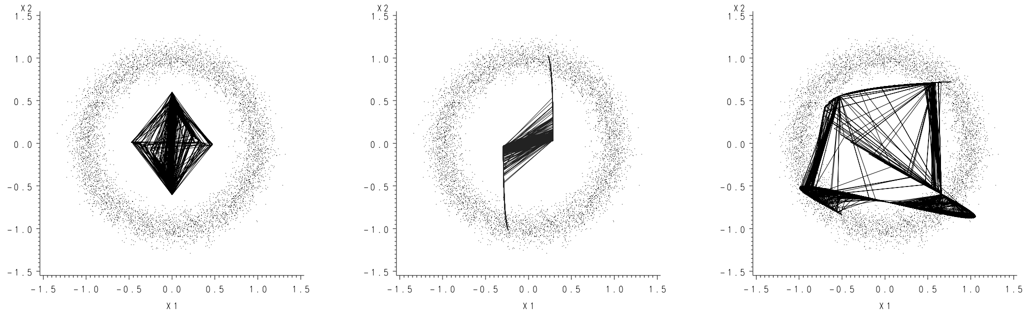


Figure 6: *Estimated circles with 6 nodes in 2nd and 4th layer.*

layer and a sigmoid activation function, the value of this node will always be positive. So the sign of the weight between this last hidden node and the output node determines the sign of the output value. This means, when the net is trained and is used for predicting values, the sign of the predicted value is determined and is independent of the signs or values of the input values. When training a network for predicting both, positive and negative values, at least one more node in the last hidden layer is needed, so that there are weights with both types of signs. Therefore we assumed, that the number of nodes in the last hidden layer depends also on, whether values with both types of signs are predicted or not. Therefore, we shifted the semicircle by adding a constant value to $x_2$. We found, that this added value increases the number of nodes needed for successful estimation to $A=4$ and $C=4$ nodes. It seemed, that this additional node was needed for modelling an estimate of the bias.

## 5.2 Estimation of circles

In literature the ability to estimate a circle is often described as the ability to estimate highly nonlinear structures. Thus we used 6284 data points from a unit-speed-circle, again with standard normal random errors $e_i$ added to both coordinates, $x_{1i} = cos(i) + e_i/10$ and $x_{2i} = sin(i) + e_i/10$ with $i \epsilon [-\pi, \pi]$in

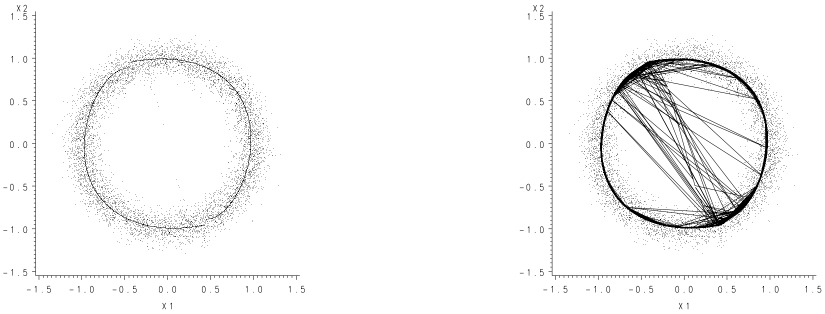


Figure 7: *Estimated circle with 6 nodes in 2nd and 4th layer each using apriori information.*

steps by 0.001. Estimating this circle with help of a neural network with 6 nodes in each, the 2nd and 4th layer, the estimates were most of the times looking like Figure 6. Although it was much easier to estimate the semicircle with more nodes, increasing the number of nodes to 7 or 9 in 2nd and 4th layer did not improve the estimation of the circle or prevent the network from converging into local minima in most of the runs.

Using the additional information about the "underlying true" scores and estimating the two subnets separately, 6 nodes in each the 2nd and the 4th layer are sufficient for a quite successful estimation (Figure 7, left). The net gives a good estimate of the functions $f$ and $s_f$. The estimated points are laying in the center of the cloud of points. While the points on the circle are estimated appropriately, no clear order with respect to their sequence

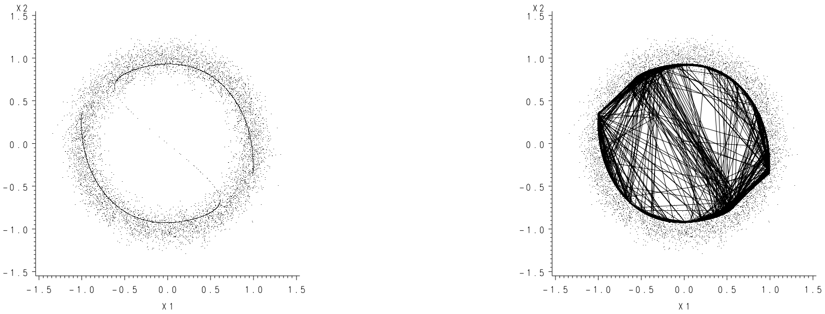


Figure 8: *Estimated circle with 7 nodes in 2nd and 4th layer each using apriori information.*

can be seen. Even though the data points are given in ordered sequence, the procedure does not build up the circle in a clockwise or counterclock wise manner, but seems to distribute on the circle line quite arbitrarily. This is visualized in Figure 7 (right) by connecting points estimated immediately after each other. This means, that quite equal data points from predictive variables result in quite different dependent values. This is obvious from the lines crossing the circle. Using a network with 7 nodes in the 2nd and 4th layer each, we could not find starting values, for which the goodness-of-fit $R^2$ could have been improved, compared to using 6 nodes each( see Table 1). However, the order in which the points on the circle are estimated remains random and the phenomenon described above is expressed more extensively (Figure 8, right).

Malthouse (1998) describes difficulties in estimating a circle in the part, where the circle is closing [$\pi$ to 0]. We could not confirm this result for these values, but found for both circles signs of a line crossing the circle (see Figures 7, left and 8, left).

Malthouse also pointed out, that in Kramers approach to NLPCA the continuity of the projection-index $s_f$ often results in problems in estimating functions or relationships with many ambiguous points. These so-called ambiguous points are data points for which the distance to two or more points on the curve $\mathbf{f}$ is equally far. By definition of the projection-index, $s_f$ is restricted to project points to the location on $\mathbf{f}$, that gives the highest score. But on the other side, by estimating $s_f$ with help of neural networks, $s_f$ is restricted to a continuous function, which might be an explanation for the line crossing the circle.

For both estimates of the circle (Figure 7 and 8), we took a closer look at the estimated scores given in the bottleneck. While the actual scores range

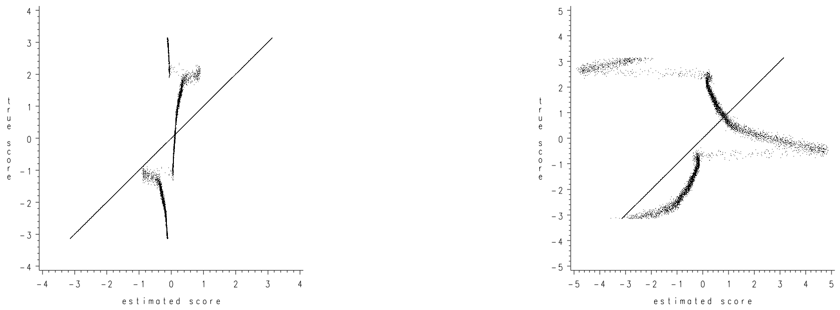


Figure 9: *Estimated scores in comparison with true scores for the circle with 6 nodes in 2nd and 4th layer each (left) and 7 nodes in 2nd and 4th layer each (right).*

from $-\pi$ to $\pi$, the estimated scores for both estimations of the circle show a much smaller range for 6 nodes, but not for 7 nodes. Figure 9 reveals no obvious relationship between the actual and estimated score.

Even with an increased number of nodes, estimating the circle without prior information about the score is difficult. So, one of the main problems in solving these nonlinear equation system is to find good starting values for parameter estimation. In many programs written for the use of neural networks e.g. Stuttgarter Neuronale Netze Simulator (1995), starting values are chosen randomly. This means, that good parameter estimates are found more or less by chance, leaving broad space for fruitless runs.

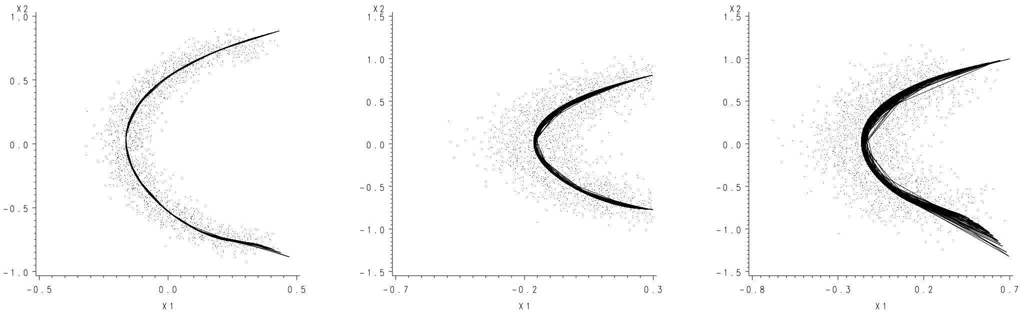


Figure 10: *Estimated unit-speed curve tested on data sets with different error sizes added.*

## 5.3 Estimation of curves

In this last simulation we again generated 1981 data points from unit-speed curves. We chose functions for $x_1$ and $x_2$ to fulfill the unit-speed property given in Equation (4.3) with $x_{1i} = 0.5i^2 - 0.16 + 0.05\, e_i$ and $x_{2i} = 0.5\, i\sqrt{1-i^2}$ $+ 0.5 artan(i) + 0.05\, e_i$ with standard normal random errors $e_i$ added to both and $i \,\epsilon\, [-0.99, 0.99]$ again in steps by 0.001. Since a neural network with 6 nodes in the 2nd and 4th layer each seemed sufficient for estimating a circle, it is used for this model also.

Basically an ideal case of a practical application is simulated, in which the net is trained to estimate the functions $s_f$ and $\mathbf{f}$ with help of a 'learning data set' and then applied on data sets, so-called 'test data sets', generated from the same model but with different random error terms added. Therefore it becomes possible to find out, whether the network is trained to distinguish between pattern and random errors in the data. For the first test data set random errors of the size as for the training set are added (Figure 10, left). In addition to this two more data sets are generated from the model, but with larger error terms added: $0.1\, e$ (Figure 10, middle) and $0.15\, e$ (Figure 10, right). Looking only at the shape of the estimated curve it is seen that in both cases the structure can be estimated out of the predictive variables quite well. But it is also obvious, that with increasing error terms the order, in which the data points are estimated becomes more random.

# 6 Discussion

After a review on the theory of linear and nonlinear principal component analysis performed with help of neural networks we investigated the performance of Kramer's (1991) nonlinear principal component analysis by means of simulation studies. We estimated circles and semicircles and by a close examination of the scores we got an insight in the characteristics of 'bottleneck'-networks used for the NLPCA.

We derived first values for the size of neural networks, i.e. the number of nodes in the hidden layers required for successfully estimating the functions $\mathbf{f}$ and $s_f$.

In our simulation studies we treated neural networks as systems of nonlinear equations. The attempt to restrict the estimation of $\mathbf{f}$ to unit-speed functions explicity by adding an additional equation failed. So, one problem of these simulations is that the estimated functions are not of unit speed. When using a restriction in form of a penalty term as suggested by Malthouse (see Equation 4.3) the restriction on unit-speed is not always accomplished. Since theory relies on the unit speed assumption, it would be a major improvement if solutions of nonlinear systems could be restricted to functions fulfilling the unit-speed property exactly instead of only penalising functions diverging to strongly.

Although our simulations give a first insight in the characteristics of 'bottleneck'-networks used for the NLPCA, additional simulation studies are needed to verify and extend our results.

## Acknowledgements

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of complexity in multivariate data structures") is gratefully acknowledged. I particularly thank Prof. Weihs for helpful discussions and his constructive criticism.

## References

BRONSTEIN, I/ SEMENDJAJEW, K. (1979) *Taschenbuch der Mathematik* Teubner Verlagsgesellschafft, Leibzig, 24. Auflage.

CYBENKO, G. (1989). *Approximation by Superpositions of a Sigmoidal Function.* Mathematics of Control, Signals and Systems (**2**), p. 303-314.

HASTIE, T.(1984). *Principal Curves and Surfaces.* Technical Report No. 11, Department of Statistics, Stanford University.

JOHNSON, R.A./ WICHERN, D.W. (1992). *Applied Multivariate Statistical Analysis.* 3rd edition Prentice Hall, New Jersey

KRAMER, M. (1991). *Nonlinear principal component analysis using autoassociative neural networks.* AIChE Journal (**37**), p. 233-243.

MARDIA, K./ KENT, J./ BIBBY, J. (1979). *Multivariate Analysis.* Academic Press, London.

MALTHOUSE, E.C./ TAMHANE, A.C./ MAH, R.S.H. (1995A). *Some Theoretical results on Nonlinear Principal Components Analysis.* Proceedings of the American Control Conference, p. 744-748.

MALTHOUSE, E.C. (1995). *Nonlinear Partial Least Squares.* Dissertation, Northwestern University/Illinois.

MALTHOUSE, E.C. (1998). *Limitation of Nonlinear PCA as Performed with Generic Neural Networks.* IEEE Transactions on Neural Networks (**9**), p. 165-173.

SANGER, T. (1989).*Optimal unsupervised learning in a single-layer linear feedforward neural network.* Neural networks (**2**), p. 459-473.

SCHMIDLI, H. (1995). *Reduced Rank Regression.* Physica Verlag, Heidelberg.

STUTTGARTER NEURAL NETWORK SIMULATOR (1995). *http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html.*

THORPE, J. (1979). *Elementary Topics in Differential Geometry.* Springer-Verlag, New York.