Some Theoretical Results on Nonlinear Principal Components Analysis

Edward C. Malthouse

September 19, 1996

Abstract

Nonlinear principal components analysis (NLPCA) neural networks are feedforward autoassociative networks with five layers. The third layer has fewer nodes than the input or output layers. NLPCA has been shown to give better solutions to several feature extraction problems than existing methods, but very little is know about the theoretical properties of this method or its estimates. This paper studies NLPCA. It proposes a geometric interpretation by showing that NLPCA fits a lower-dimensional curve or surface through the training data. The first three layers project observations onto the curve or surface giving scores. The last three layers define the curve or surface. The first three layers are a continuous function, which I show has several implications: NLPCA "projections" are suboptimal producing larger approximation error, NLPCA is unable to model curves and surfaces that intersect themselves, and NLPCA cannot parameterize curves with parameterizations having discontinuous jumps. I establish results on the identification of score values and discuss their implications on interpreting score values. I discuss the relationship between NLPCA and principal curves and surfaces, another nonlinear feature extraction method.

Keywords: nonlinear principal components analysis, feature extraction, data compression, principal curves, principal surfaces.

1 Introduction

Nonlinear principal components analysis (NLPCA) [11] is one approach to nonlinear feature extraction. It uses five-layer autoassociative neural networks with a bottleneck layer of nodes (e.g., Fig. 1) to reduce the dimension of the input variables. The second and fourth layers of the network have sigmoidal activation functions, so layers 1, 2, and 3 and layers 3, 4, and 5 model nonlinear functions. The third layer has fewer nodes than the first or fifth. The values of the output nodes in layer 5 are trained to approximate the inputs. After the network has been trained, the bottleneck node activation values in layer 3 give a lower dimensional representation of the inputs.

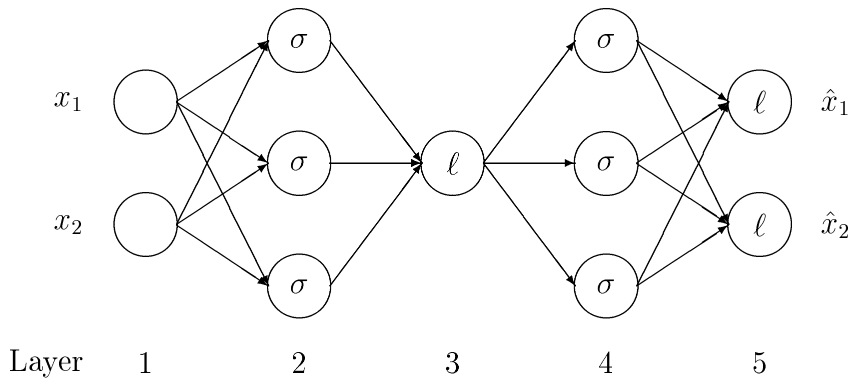


Figure 1: NLPCA neural network architecture. Linear activiation function are denoted by ℓ and sigmoidal functions by σ.

NLPCA and related nonlinear feature extraction methods have been successfully applied to many different problems. For example, [4] shows how NLPCA can be applied to image compression problems. [17] show how a version of NLPCA can be used for another data compression problem. [5] shows how another version of NLPCA can be used to construct control charts. These examples suggest that NLPCA is a promising method. But very little is known about the theoretical properties of NLPCA and its estimates.

The purpose of this paper is to investigate the properties of the NLPCA estimates. Section 2 proposes a geometric interpretation for NLPCA. NLPCA reduces the dimension of the inputs by fitting a curve or surface through the data. The first three layers (1, 2, and 3) of the network project the original data onto the curve or surface and the activation values of the bottleneck node, called *scores*, give the location of the projection. The last three layers

(3, 4, and 5) define the curve or surface. This approach is similar to principal curves and surfaces, a different nonlinear feature extraction method. Layers 1, 2, and 3 model only continuous functions and section 3 examines the implications of this fact. First, the projections onto (nonlinear) curves and surfaces are suboptimal in that the inputs are not mapped to point on the curve or surface that is closest; this increases training error and leads to incorrect results. Second, NLPCA cannot model curves or surfaces that intersect themselves, for example circles. Third, NLPCA cannot parameterize curves or surfaces with parameterizations that have discontinuous jumps. Section 4 examines the identification of score values. The score values depend on how the curve or surface is parameterized, which for NLPCA is determined by the starting weight values and the nonlinear optimization routine. I show that the score values are identified to a homeomorphic transformation and discuss the implications of this result on interpreting score values. I summarize the conclusions and give directions for future research in Section 5.

2 Feature Extraction Methods

This section gives an overview of the feature extraction problem and three approaches to solving it. Let $\mathbf{X}$ be an $n \times p$ matrix that contains n observations (subjects, cases, items, etc.) on p variables (process variables, attributes, etc.). Denote the i^{th} row vector of $\mathbf{X}$ by $\mathbf{x}_i(p \times 1)$ and the j^{th} column vector by $\mathbf{x}_{(j)}(n \times 1)$. Without loss of generality, assume that the columns of $\mathbf{X}$ are mean centered, i.e.,

$$\mathbf{x}_{(j)} \leftarrow \mathbf{x}_{(j)} - \text{average}(\mathbf{x}_{(j)}),$$

where

$$\text{average}(\mathbf{x}_{(j)}) = \frac{1}{n}\sum_{i=1}^{n} x_{ij}.$$

The superficial dimension of the observations is p, but there are often dependencies among the columns of $\mathbf{X}$, and the intrinsic dimension of the observations, labeled by r, can be much smaller than p. In these situations, one may want to extract a new set of variables called *features* that contain the same information as $\mathbf{X}$, but have the smaller intrinsic dimension. I call the values of these feature variables *scores* and denote them by $n \times r$ matrix $\mathbf{S}$.

The observations and scores are hypothesized to be related as follows:

$$\mathbf{x}_i = \mathbf{f}(\mathbf{s}_i) + \boldsymbol{\epsilon}_i = \begin{pmatrix} f_1(\mathbf{s}_i) \\ \vdots \\ f_p(\mathbf{s}_i) \end{pmatrix} + \begin{pmatrix} \epsilon_{i1} \\ \vdots \\ \epsilon_{ip} \end{pmatrix}, \tag{1}$$

$\boldsymbol{\epsilon}_i(p \times 1)$ is vector of noise and $\mathbf{f} : \mathcal{R}^p \to \mathcal{R}^p$ is a smooth function. The feature extraction problem is to find estimates for $\mathbf{S}$ and $\mathbf{f}$.

Function $\mathbf{f}$ is called a *(globally) parameterized r-surface* in $\mathcal{R}^p$. Vector $\mathbf{s}$ describes the location of point $\mathbf{x}$ relative to the parameterization of surface $\mathbf{f}$. When $\mathbf{s}$ is unidimensional, surface $\mathbf{f}$ is called a *curve*. For example, the unit circle, $\{(x_1, x_2) \in \mathcal{R}^2 : x_1^2 + x_2^2 = 1\}$, is an example of a one-dimensional parameterized curve in $\mathcal{R}^2$ with

$$\mathbf{x} = \mathbf{f}(s) = \begin{pmatrix} \cos s \\ \sin s \end{pmatrix} \tag{2}$$

and $s \in [0, 2\pi)$. The definition of a curve/surface is not unique and there are many different functions $\mathbf{f}$ that define the same curve/surface. For example, the unit circle can also be defined by

$$\mathbf{x} = \mathbf{f}^*(s) = \begin{pmatrix} \cos(as) \\ \sin(as) \end{pmatrix}, \tag{3}$$

where a is a constant and $s \in [0, 2\pi/a)$. Curve $\mathbf{f}^*$ defines the same curve as the original $\mathbf{f}$ in (2), but with a different parameterization, i.e., $\mathbf{f}(s) = \mathbf{f}^*(s/a)$.

Curves are often parameterized by arc length, i.e., $\mathbf{f}(s_1) - \mathbf{f}(s_0)$ is the length along curve $\mathbf{f}$ from s_0 to s_1. Using calculus, the arc length of curve $\mathbf{f}$ from s_0 to s_1 is given by

$$\int_{s_0}^{s_1} \sqrt{\sum_{i=1}^{p} \Big(\frac{df_i}{ds}\Big)^2}\, ds = s_1 - s_0.$$

Curve $\mathbf{f}$ is parameterized by arc length if and only if it has the *unit-speed* property defined by

$$\sqrt{\sum_{i=1}^{p} \Big(\frac{df_i}{ds}\Big)^2} = 1.$$

From differential geometry, every smooth curve can be parameterized by arc length. For example, any vector with unit length has the unit-speed property.

The unit circle $\mathbf{f}$ in (2) has unit speed, but the reparameterized unit circle $\mathbf{f}^*$ in (3) does not. Every curve (one-dimensional surface) can be parameterized by arc length and this parameterization is unique to choice of origin and sign flips, but defining parameterizations for higher-dimensional surfaces is much more complicated. The unit-speed property can be generalized for surfaces in terms of areas and volumes, but these parameterizations are not unique. See [8] for further discussion.

2.1 Principal Components Analysis

Principal components analysis is a well-established feature extraction method that assumes $\mathbf{f}$ in (1) is linear. Let the spectral decomposition of $\mathbf{X}'\mathbf{X}$ be $\mathbf{X}'\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}'$, where $\mathbf{U}(p \times p)$ is an orthogonal matrix whose column vectors are unit-length eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal elements, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, are eigenvalues. The *principal component transformation* is given by

$$\mathbf{S} = \mathbf{X}\mathbf{U}, \tag{4}$$

where $\mathbf{S}(n \times p)$ is a matrix of score values. The PCA transformation has several important properties (e.g., see [13]).

1. The principal component transformation is a change of basis from the basis of "unit vectors" $\mathbf{I}$ to a basis consisting of the column vectors of $\mathbf{U}$. Column vector $\mathbf{u}_{(j)}(p \times 1)$ is called the j^{th} *principal axis* or *principal direction.*

2. The total variance in $\mathbf{X}$ (with mean-centered columns) is

$$\sum_{i=1}^{n}\sum_{j=1}^{p} x_{ij}^2 = \operatorname{tr}(\mathbf{X}'\mathbf{X}) = \operatorname{tr}(\mathbf{G}\Lambda\mathbf{G}') = \sum_{j=1}^{p} \lambda_j, \tag{5}$$

   and the variance along the j^{th} principal axis is

$$\operatorname{var}(\mathbf{x}_{(j)}) = (\mathbf{X}\mathbf{g}_{(j)})'\mathbf{X}\mathbf{g}_{(j)} = \mathbf{g}'_{(j)}\mathbf{G}\Lambda\mathbf{G}'\mathbf{g}_{(j)} = \lambda_j. \tag{6}$$

3. No standardized linear combination of the rows of $\mathbf{X}$ has larger variance than λ_1, i.e.,

$$\lambda_1 = \max_{\mathbf{a}'\mathbf{a}=1} \operatorname{var}(\mathbf{X}\mathbf{a}) = \operatorname{var}(\mathbf{X}\mathbf{u}_{(1)}). \tag{7}$$

   Likewise, if $\mathbf{a}$ is also constrained to be orthogonal to $\mathbf{u}_{(1)}, \ldots, \mathbf{u}_{(k-1)}$, then the variance in (7) has a maximum value of λ_k.

4. The subspace spanned by the first k eigenvectors has smaller mean square deviations from the original $\mathbf{X}$ matrix than any other subspace with dimension k, i.e., if the columns of $\mathbf{A}(p \times k)$ are a basis for a subspace in $\mathcal{R}^p$, then

$$\min_{\mathbf{A}} \|\mathbf{X} - \text{proj}_{\mathbf{A}}\mathbf{X}\| = \|\mathbf{X} - \text{proj}_{\mathbf{U}_{(k)}}\mathbf{X}\|, \tag{8}$$

where $\text{proj}_{\mathbf{A}}\mathbf{X}$ denotes the projection of the row vectors of $\mathbf{X}$ onto the subspace spanned by the columns of $\mathbf{A}$.

NLPCA generalizes the objective function given in (8). There are several methods that generalize (7) including [6], [7], [10], and [15]. Also see Section 10 in [12] for additional discussion and references on these other approaches.

2.2 Principal Curves and Surfaces

Principal curves (PC) were first proposed in [8] and [9]. PCA finds a unit-length vector $\mathbf{u}$ that satisfies the minimum distance property in (8). PC extends PCA by fitting a unit-speed curve $\mathbf{f}$ under a similar objective function. The lengths of the projections of $\mathbf{X}$ onto PCA vector $\mathbf{u}$ were given in (4); the principal curves method generalizes this expression by defining a *projection index* $s_{\mathbf{f}}$ to map observations in $\mathcal{R}^p$ to a point on curve $\mathbf{f}$ that is closest to it:

$$s_{\mathbf{f}}(\mathbf{x}) = \sup_{s}\{s : \|\mathbf{x} - \mathbf{f}(s)\| = \inf_{\mu} \|\mathbf{x} - \mathbf{f}(\mu)\|\}. \tag{9}$$

The projection index $s_{\mathbf{f}}(\mathbf{x})$ evaluated at $\mathbf{x}$ gives the arc length along $\mathbf{f}$ from some fixed origin to the point that is closest to $\mathbf{x}$. If there are several such points, by convention $s_{\mathbf{f}}$ selects the one with the largest score value. Points that can be projected to more than one point on the curve are called *ambiguity points*. Fig. 2 illustrates the role of $s_{\mathbf{f}}$. Curve $\mathbf{f}$ has a parabolic shape, for example[1]

$$\mathbf{f}(s) = \begin{pmatrix} s \\ s^2 \end{pmatrix}. \tag{10}$$

Observation $\mathbf{x}_i$ is equally close to both branches of the parabola and the projections onto both sides are shown. The projection index $s_{\mathbf{f}}$ selects the

[1]Note that this definition of a parabola does *not* have the unit-speed property. I make this simplification to clarify the discussion.

projection onto the right branch because its score value (the arc length from the origin, $s = 0$) is greater, assuming that $\mathbf{f}$ is parameterized so that s increases with x_1.

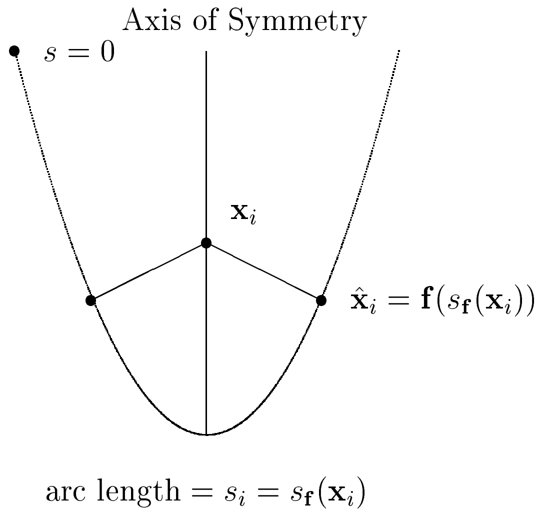


Figure 2: Graphical illustration of the projection index

The principal curves method estimates $\mathbf{f}$ under the following least-squares objective function[2]:

$$\min_{\mathbf{f}} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{f}(s_{\mathbf{f}}(\mathbf{x}_i))||^2. \tag{11}$$

The composition of functions $\mathbf{f}(s_{\mathbf{f}}(\mathbf{x}_i))$ gives the p-dimensional coordinates of the projection of $\mathbf{x}_i$ onto curve $\mathbf{f}$. The objective function is minimized with the principal curve algorithm, which alternates between estimating $\mathbf{f}$ with $\mathbf{s}$ fixed and estimating $\mathbf{s}$ with $\mathbf{f}$ fixed. When $\mathbf{f}$ is defined to be linear, the principal curves algorithm is equivalent to the power method of extracting the dominant eigenvalue from $\mathbf{X}'\mathbf{X}$, and therefore extracts the first principal component.

[2]More formally, let X be a random vector defined on $\mathcal{R}^p$ from continuous probability density ψ with $E(X) = \mathbf{0}$. [9] defines a principal curve of ψ to be the set of curves that do not intersect themselves and are *self-consistent*, i.e., $E(X|s_{\mathbf{f}}(X) = s) = \mathbf{f}(s)$, for all s. A curve is self-consistent if each point (call it $\mathbf{f}(s)$) is the mean of all points in the support of ψ that are projected on s. Under this definition of principal curves, they show that a curve is a principal curve if and only if it satisfies the objective function in (11).

Two-dimensional principal surfaces (PS) were proposed in [8] and were later extended in [12] for higher dimensional surfaces. Principal surfaces extend PCs by fitting a surface that satisfies a minimum distance objective function similar to (11). The projection index for a two-dimensional surface $\mathbf{f}$ is

$$\mathbf{s_f}(\mathbf{x}) = \sup_{s_2} \sup_{s_1} \{\mathbf{s} : \|\mathbf{x} - \mathbf{f}(\mathbf{s})\| = \inf_{\boldsymbol{\mu}} \|\mathbf{x} - \mathbf{f}(\boldsymbol{\mu})\|\}.$$

Surface $\mathbf{f}$ is estimated under the objective function

$$\min_{\mathbf{f}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{f}(\mathbf{s_f}(\mathbf{x}_i))\|^2$$

using the principal surface algorithm, which is similar to the principal curve algorithm.

Finding a parameterization for an r-dimensional surface is much more difficult than finding one for a curve. If $\mathbf{f}$ is a *curve* and one assumes it is continuous, Result 1 discussed below shows that the parameterization is determined down to choice origin and scaling; principal curves are *defined* to have unit speed and the origin of a PC is *defined* to be one of the two end-points of the data. But the indeterminacy is more problematic with surfaces due to other factors like rotations. For example, consider parameterizing a hemisphere, $\{(x_1, x_2, x_3) \in \mathcal{R} \times \mathcal{R} \times \mathcal{R}^+ : x_1^2 + x_2^2 + x_3^2 = 1\}$. The usual spherical coordinates of vector $\mathbf{x}$ lying on this sphere are (s_1, s_2), where s_1 is the angle made between the x_1 axis and the projection of $\mathbf{x}$ onto the x_1-x_2 plane (latitude) and s_2 is the angle between $\mathbf{x}$ and the x_3 axis (longitude). The relationship between $\mathbf{x}$ and $(s_1, s_2) \in [0, 2\pi) \times [0, \pi/2)$ is

$$\mathbf{x} = \mathbf{f}(s_1, s_2) = \begin{pmatrix} \sin s_2 \cos s_1 \\ \sin s_2 \sin s_1 \\ \cos s_2 \end{pmatrix}. \tag{12}$$

There are many other ways of parameterizing this hemisphere. Alternatively, the hemisphere could be parameterized by projecting each $\mathbf{x}$ onto the x_1-x_2 plane and setting $(s_1, s_2) = (x_1, x_2)$, i.e., $\mathbf{f}(s_1, s_2) = (s_1, s_2, \sqrt{1 - s_1^2 - s_2^2})'$. A third parameterization is to use *stereographic projections*. The stereographic projection of a point $\mathbf{x}$ onto the x_1-x_2 plane is given by the point of intersection between the x_1-x_2 plane and the line passing through the "North pole" $(0, 0, 1)$ and $\mathbf{x}$; the stereographic projection of $(0, 0, 1)$ onto the x_1-x_2 plane is undefined. [12] uses the second approach. A problem with this approach is that surfaces that bend back on themselves like a full sphere[3] cannot be parameterized this way. For example, a 3/4 sphere (Equation 12 with $s_1 \in [0, 2\pi)$ and $s_2 \in [0, 3\pi/4])$ cannot be parameterized in this way, but it could be parameterized with the spherical coordinate or stereographic projection approach. Example 6.4 in [8] uses stereographic projections to parameterize a full sphere. I revisit this discussion in section 3.2 and show that the spherical coordinates for this hemisphere as defined in (12) cannot be used as a parameterization if the projection index is continuous.

Principal curves are defined to have unit speed. To address the issue of scaling for principal surfaces, [12] suggests rescaling each score vector $\mathbf{s}_{(j)}$ so that $0 \leq s_{ij} \leq 1$ with $\min_{\mathbf{s}_{(j)}} = 0$ and $\max_{\mathbf{s}_{(j)}} = 1$.

2.3 Nonlinear Principal Components Analysis

Nonlinear principal components analysis (NLPCA) [11] was proposed independently of principal curves and surfaces and was motivated by a PCA implementation using feedforward neural networks. Before presenting the NLPCA method, I first sketch the neural-network implementation of PCA. Feedforward neural networks can be used to extract principal components with the architecture shown in Fig. 3. The network has three layers, with p nodes in the input and output layers and 1 node in the hidden layer. Weights (u_1, u_2, u_3) are determined to minimize the following least-squares objective function:

$$\min_{\mathbf{u}} \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - \hat{x}_{ij})^2 = \min_{\mathbf{u}} \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - (\mathbf{u}'\mathbf{x}_i) u_p)^2. \qquad (13)$$

The network is called an *autoassociative* neural network because it is trained to reproduce its inputs. The hidden layer in an autoassociative network is also called a *bottleneck layer* because the p-dimensional inputs must pass through the r-dimensional bottleneck layer before reproducing the inputs. Data compression therefore occurs in the bottleneck layer. Note that this architecture estimates the PCA solution because it minimizes the same objective function as PCA (Equation 8). An early description of this application of neural networks is the encoder-decoder problem described in [14, pp. 335–339]. [1] later develops some theoretical properties of this method.

[3] The statement is true for any partial sphere "greater" than a hemisphere, e.g., Equation (12) with $s_1 \in [0, 2\pi)$, $s_2 \in [0, a)$, and $a > \pi/2$.

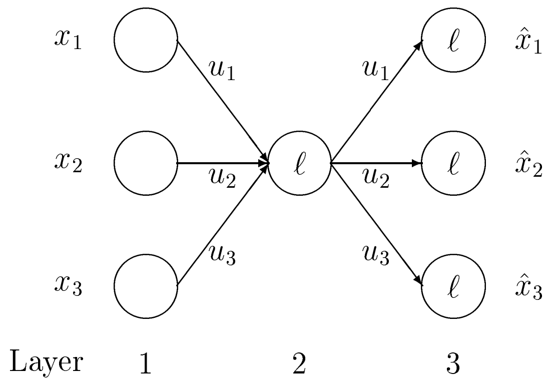


Figure 3: Neural network for principal components analysis. Linear activation functions denoted by ℓ.

NLPCA is a direct generalization of the neural network implementation of PCA. A three-layer neural network with nonlinear activation functions in the hidden layer can represent any *continuous* function under weak assumptions [3]. NLPCA adds hidden layers with nonlinear activation functions between the input and bottleneck layers and between the bottleneck and output layers giving a network with a total of 5 layers. The network models a composition of functions. Fig. 1 shows an example of an NLPCA network. The five-layer NLPCA network has p nodes in the input layer, r nodes in the third (bottleneck) layer, and p nodes in the output layer. The nodes in layers 2 and 4 must have nonlinear activation functions so that layers 1, 2, and 3 and layers 3, 4, and 5 can represent arbitrary smooth functions. The nodes in layers 3 and 5 usually have linear activation functions, although they could be nonlinear. Direct connections are allowed between layers 1 and 3 and between 3 and 5, but direct connections are not allowed to cross bottleneck layer 3. As with the PCA networks, data compression takes place because the p-dimensional inputs must pass through the r-dimensional bottleneck layer before reproducing the inputs. Once the network has been trained, the bottleneck node activation values give the scores.

I now define some notation that will facilitate the discussion of this method and its relationship to principal curves and surfaces. Since three-layer neural networks with nonlinear activations functions are continuous functions, the subnetworks consisting of layers 1, 2, and 3 and layers 3, 4, and 5 are *continuous* functions. Let $\mathbf{s_f} : \mathcal{R}^p \to \mathcal{R}^r$ denote the function modeled by layers 1, 2, and 3 and let $\mathbf{f} : \mathcal{R}^r \to \mathcal{R}^p$ denote the function modeled by layers 3, 4, and 5. Using this notation, note that the weights in the autoassociative NLPCA network are determined under the following objective function:

$$\min_{\mathbf{f}, s_{\mathbf{f}}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{f}(s_{\mathbf{f}}(\mathbf{x}_i))\|^2. \tag{14}$$

Principal curves and NLPCA (with one bottleneck node, $r = 1$) have the following features in common:

1. Both give the PCA solution when $s_{\mathbf{f}}$ and $\mathbf{f}$ are linear.

2. Both define a function $\mathbf{f} : \mathcal{R}^1 \to \mathcal{R}^p$, a curve in $\mathcal{R}^p$.

3. Both define a function $s_{\mathbf{f}} : \mathcal{R}^p \to \mathcal{R}^1$. An important difference is that NLPCA defines $s_{\mathbf{f}}$ to be continuous whereas the principal curves projection index can be discontinuous. Section 3 discusses the effects of this difference.

4. The objective functions in Equations (11) and (14) minimize the same expression. NLPCA minimizes the expression over functions $s_{\mathbf{f}}$ and $\mathbf{f}$ while principal curves performs the minimization only over $\mathbf{f}$ and "plugs in" an optimal $s_{\mathbf{f}}$. I conjecture that if the NLPCA $s_{\mathbf{f}}$ could be *discontinuous*, the NLPCA solutions would be principal curves. Furthermore, if a principal curve were unique (and $\mathbf{f}$ were one-to-one), the principal curve and NLPCA composition of functions $\mathbf{f}(s_{\mathbf{f}}(\mathbf{x}))$ would be equal for almost every[4] $\mathbf{x}$; the solutions would therefore differ only by their parameterizations.

Similar analogies can be made between principal surfaces and NLPCA with multiple bottleneck nodes.

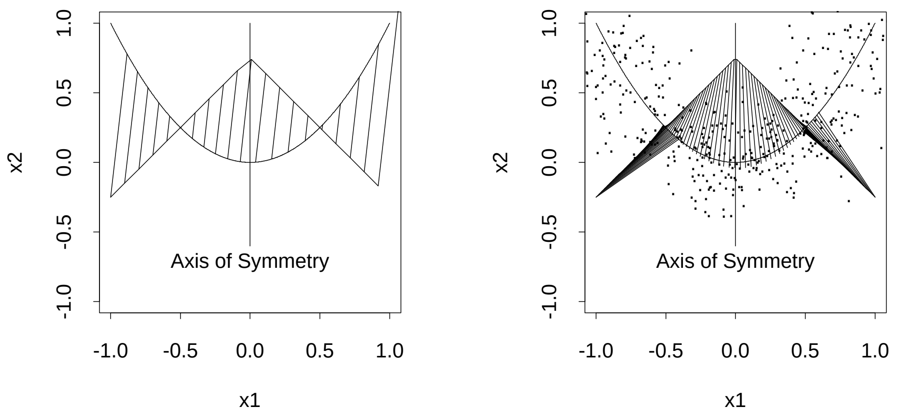
3 The Effects of a Continuous Projection Index

3.1 Suboptimal "Projections"

Defining $s_{\mathbf{f}}$ to be continuous (for all values in $\mathcal{R}^p$) causes suboptimal "projections" for certain $\mathbf{x}$ values when the curve is nonlinear. A "projection" is

[4] They would not necessarily be equal for points in the set of ambiguity points, which have measure zero [8].

suboptimal when an $\mathbf{x}$ is mapped to a point on the curve other than the point that is closest to it (the "inf" part of Equation 9). I illustrate the reason for this with an example. Suppose we are estimating the parabola given in (10) shown in Figures 4a and 4b. The set of points $\{\mathbf{x} : x_1 = 0 \text{ and } x_2 > 0\}$ (the axis of symmetry) are the ambiguity points of this curve because there is more than one point on the parabola that is closest to each one of them. If we approach an ambiguity point in the direction of a normal through the curve, an optimal $s_\mathbf{f}$ must be discontinuous when we cross the ambiguity point. The plots in Fig. 4 show normals drawn through the points $(0.5, 0.25)$ and $(-0.5, 0.25)$ that intersect at ambiguity point $(0, 0.75)$. An optimal $s_\mathbf{f}$ would project[5] each point on the normals to either $(0.5, 0.25)$ or $(-0.5, 0.25)$.



(a) Without noise

(b) With noise

Figure 4: NLPCA fit of parabola

The NLPCA model of this parabola problem was fitted using the `nn` neural network simulator written by the author[6]. The lines in Fig. 4 show where the model projects points along the normals. Fig. 4a shows the fit to

[5]More precisely, $\mathbf{f}(s_\mathbf{f}(\mathbf{x})) = (0.5, 0.25)'$ for all $\mathbf{x} \in \{\mathbf{x} : x_1 > 0 \text{ and } (x_2 - 0.25)/(x_1 - 0.5) = -1\}$.

[6]Software available from `mkt2715.kellogg.nwu.edu`.

noiseless data. NLPCA models the curve almost perfectly (training FVU[7] ≈ 0), but the projections of points that do not lie on the parabola are all wrong. Instead of mapping the points along the normals to $(0.5, 0.25)$, NLPCA maps them to the point on the curve with roughly the same x_1 coordinate. The approximation is different when noise is added to the data (Fig. 4b). The training FVU was 0.0775 for this network. The projections are good for points below the parabola ($\mathbf{x}$ such that $x_2 < x_1^2$) and for other points that are fairly close to the curve. As we approach an ambiguity point, however, the plot shows a suboptimal "fanning behavior" culminating with the ambiguity point projected (suboptimally) around $(0, 0)$. The reason for the fanning behavior is that NLPCA must avoid being discontinuous at the ambiguity point. Fig. 5 shows the principal curve fit of the parabola from noisy data[8]. Function $s_{\mathbf{f}}$ is clearly discontinuous and the projections are close to $(0.5, 0.25)$ and $(-0.5, 0.25)$.

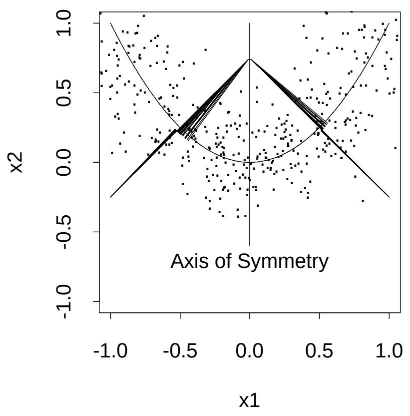


Figure 5: Principal curve fit of parabola

I also show the results of fitting a NLPCA model to a 3/4 circle, i.e., Equation 2 with $s \in [0, 3\pi/2]$. The 3/4 circle is important because it does

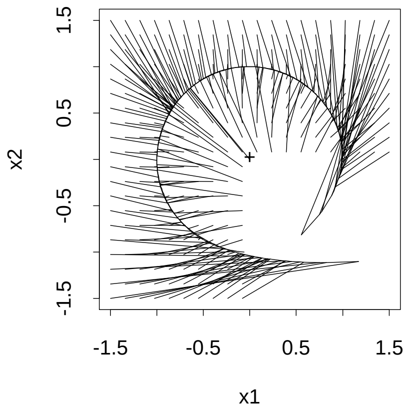
[7]FVU is the fraction of variance unexplained given by $\sum \|\mathbf{x}_i - \mathbf{f}(s_{\mathbf{f}}(\mathbf{x}_i))\|^2 / \sum \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2$, where $\bar{\mathbf{x}} = \mathbf{1}_n'\mathbf{X}/n$.

[8]Principal curves were estimated using Trevor Hastie's S implementation available via anonymous ftp from `research.att.com`.

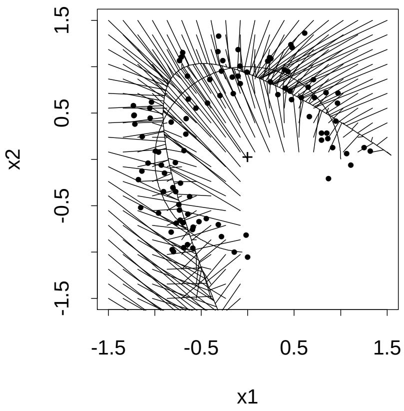
not intersect itself, yet it cannot be parameterized by "linearizing" the curve, i.e., parameterizing it with the lengths of projections onto some line, because the curve loops back on itself. The parabola and a semicircle ($s \in [0, \pi]$) can be parameterized by projections onto the x_1 axis (as in Fig. 4a) and thus PCA scores also should estimate the order of the feature correctly in noiseless situations. The center of the circle is the only ambiguity point. I generated $n = 100$ s values from $\mathcal{U}[0, 3\pi/2]$. The plots in Fig. 6 show the NLPCA fits for noiseless and noisy data. The NLPCA fit is almost perfect for the noiseless circle (training FVU = 0.0004) and the projections of points near the circle are very good. The projections of points further away from the circle are not so good. For example, the point $(-1.5, -1.5)$ is mapped near $(1.2, -1.0)$ and $(1.5, 1.5)$ is mapped near $(0.5, -0.75)$. The fanning behavior is also seen on this plot and ambiguity point (marked with "+") is projected around $3\pi/4$. Consider the projections of the vertical sequence of points from $(0.6, 0.6)$ to $(0.6, 0)$. The points near $(0.6, 0.6)$ are projected correctly, but as the points approach $x_2 = 0$ the projections are pulled toward $3\pi/4$. The pull becomes more obvious for points around $(0, 0)$.

Plot b shows the NLPCA fit for the noisy data. NLPCA appears to extract a parabolic-shaped curve instead of a $3/4$ circle and the fitted curve does not curve back on itself, as is required by a $3/4$ circle. Fig. 7 shows the principal curve fit of $3/4$ circle from noisy data. The projections are all good, but the principal curves method has some difficulty with the endpoints of the curve. [9] also notes this problem (see their Fig. 4). The projection index should map the center of the circle to one of the endpoints, depending on whether the parameterization increases in a clockwise or counter-clockwise direction. The center is mapped to the wrong point because the fit is not perfect and the points around $5\pi/4$ are closer to the center than the ends.

[8] also examines some related questions and shows that if $s_{\mathbf{f}}$ (as defined in Equation 9) is continuous at a point $\mathbf{x}$, then $\mathbf{x}$ is not an ambiguity point (Theorem 4.4). I propose a corollary to this result: if $s_{\mathbf{f}}$ is continuous, it cannot be defined at ambiguity points. Neural networks are defined for each point in $\mathcal{R}^p$ and therefore must make suboptimal projections to avoid having ambiguity points. Therefore the NLPCA solutions are not principal curves and the first three layers of an NLPCA network will make suboptimal "projections." Principal curves do not suffer from the fanning behavior. The examples in this section also suggest that an NLPCA model should not be evaluated for points that are outside the region covered by the training data (extrapolation points).



(a) Without noise



(b) With noise

Figure 6: NLPCA fit of 3/4 circle

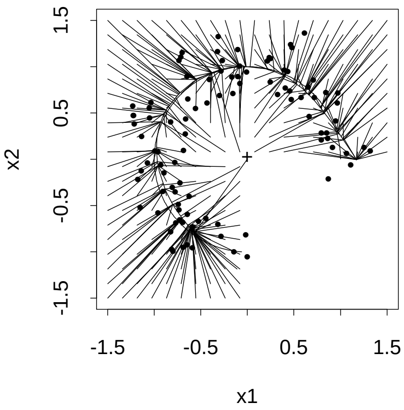
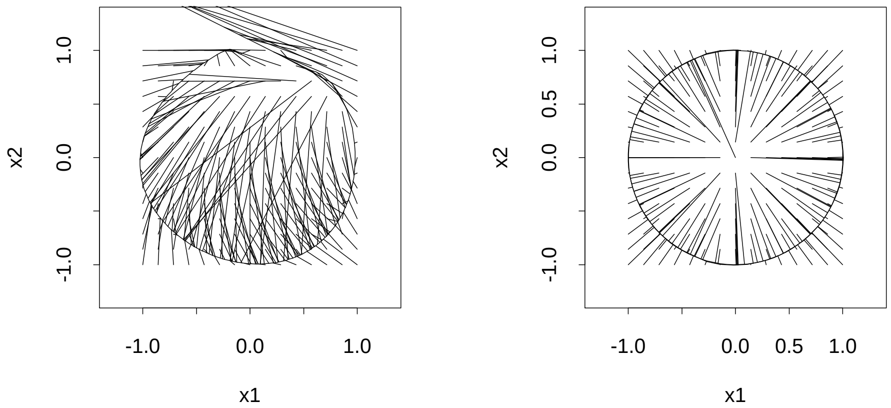


Figure 7: Principal curve fit of 3/4 circle

3.2 Reduced Class of Curves and Parameterizations

When the projection index is modeled with a continuous function, curves and surfaces that intersect themselves cannot be approximated[9] and certain, possibly desirable, parameterizations of surfaces are not possible. The reason for this can be easily understood through an example. A circle in $\mathcal{R}^2$ intersects itself and can be described by the polar coordinates in Equation (2). When $s_{\mathbf{f}}$ is defined to be continuous, a small change in $\mathbf{x}$ values must result in a small change in s values. This is clearly not the case around the point $(1, 0)$, where s jumps from 2π to 0.



(a) NLPCA

(b) Principal curve

Figure 8: NLPCA and principal curve fits of full circle

Asking NLPCA to do something it cannot do, i.e., model a curve that intersects itself, can produce some strange results, as illustrated in Fig. 8a. I generated 100 equally-spaced s values in the interval $[0, 2\pi]$ and their corresponding 100 points along the perimeter of a circle using (2). I fitted an NLPCA model and then evaluated the model on a cross-validation grid of

[9][17] independently noted that NLPCA has problems in approximating curves that intersect themselves.

points in $\mathcal{R}^2$ to understand the NLPCA solution. To insure thorough training, I trained the network for 6000 iterations with the quasi-Newton BFGS nonlinear optimization algorithm and the resulting FVU was 0.0097. Because the data were noiseless, one would have expected FVU ≈ 0. The curve in Fig. 8a is the NLPCA approximation and shows that NLPCA does a good job of extracting a circle, except around $\pi/2$, where the ends of the curve seem to repel each other. Most of the training FVU (0.0097) results from the poor fit around $\pi/2$. The "projections" on the cross-validation grid are bad, particularly in the first quadrant, where $s_{\mathbf{f}}$ becomes highly nonlinear and sprays its "projections" around the entire perimeter of the circle. The "projections" are particularly bad along the positive part of line $x_1 = 1$, where the "projections" extend off the plot. An optimal $s_{\mathbf{f}}$ would project all the points in the first quadrant to points on the circle in the first quadrant, but these "projections" are clearly suboptimal.

Fig. 8b shows the principal curve fit. The estimate is nearly perfect and almost all the projections are in the right place. One exception to this is the point $(0, 0.1)$, which is mapped to the right of $\pi/2$.

Similarly, parameterizations that have discontinuous jumps cannot be used when the projection index is continuous. For example, if the projection index is continuous, the hemisphere discussed in section 2.2 could not be parameterized by the spherical coordinates as specified in (12) because, like the circle example above, s_1 must jump from 2π to 0. Spherical coordinates could be used to parameterize a hemisphere by exchanging the x_2 and x_3 coordinates, i.e., $\mathbf{f}(s_1, s_2) = (\sin s_2 \cos s_1, \cos s_2, \sin s_2 \sin s_1)'$, $(s_1, s_2) \in (0, \pi) \times (0, \pi)$.

4 Score Interpretations

The solutions of nonlinear feature extraction methods are obtained by solving a least-squares optimization problem. Whenever there is a unique minimum to this problem, the *fitted values*, $\mathbf{f}(\mathbf{s})$, will be unique (except possibly at ambiguity points). But the *score values*, $\mathbf{s}$, are not unique because there are as many different sets of score values as there are different parameterizations of the curve or surface; for example, recall the two parameterizations of a unit circle given in Equations (2) and (3). This score indeterminacy raises an important question for applications where the score values are interpreted or used to make decisions: Would the decision or interpretation change if the curve or surface were parameterized differently? This question is particularly important since the parameterizations resulting from NLPCA depend on the starting values and the choice of nonlinear optimization method.

There are many applications of nonlinear feature extraction methods where the score values are interpreted. One application I will call latent variables. Suppose that $x_1, \ldots, x_p$ are p measurements of some variable s. Each measurement, x_j, is known to be related to s by some continuous function, but each is subject to error as in Equation (1). The function $\mathbf{f}$ and latent variable values can be estimated with a suitable feature extraction method. Latent variable problems are common in the physical, social, and engineering sciences. [2] gives a thorough account of the theory of linear latent variable models for continuous and discrete data. A second application is data visualization. When the dimension of the predictor variables is large (usually $p \geq 4$), it is difficult to visualize the data graphically. Plots of the data projected onto lower-dimensional subspaces can reveal important aspects of the data. A third application is control charts. Many variables are commonly monitored during the fabrication of a product. The goal is to use these data to determine if the fabrication process is operating normally (in control) or if something is wrong (out of control). It is often difficult to distinguish between in-control and out-of-control states when p is large. [5] demonstrates that the score values from a nonlinear feature extraction method can be plotted on a control chart to detect out-of-control conditions.

Not all applications of nonlinear feature extraction methods require interpreting scores. One application is data/image compression. The intrinsic dimension of the data can be substantially less than the superficial dimension. In this case the score values provide a compressed version of the original data. Note that the score values are not interpreted or used to make any decisions. [17, Example 4] capture 95% of the variation in $p = 65$ observed x's with $r = 3$ nonlinear components. [4] uses a version of NLPCA for image compression. The authors analyze 64×64 pixel images ($p = 4,096$) and found $r = 5$ dimensional representations of the images. A second application is noise reduction. When the observed variables are noisy, using the fitted values $\mathbf{f}(s)$ can reduce the noise. The distinction between noise reduction and latent variable applications is that in the former the fitted values $\mathbf{f}(s_{\mathbf{f}}(\mathbf{x}))$ are used while in the latter the scores $s_{\mathbf{f}}(\mathbf{x})$ are used. A third application is curve estimation. The nonlinear feature extraction methods (with a discontinuous projection index) fit a curve passing through the middle of a set of data points. [9] uses principal curves to determine the path of particles in a collider chamber.

The following result partially resolves the parameterization question. The result shows that any two parameterizations of a curve or surface satisfying the least-squares objection function in (14) can be related by a homeomorphism[10]. I prove the result in the Appendix.

Result 1 *Let* $(\mathbf{s_f}, \mathbf{f})$ *be a projection index-surface pair minimizing the least-squares objective function (14) and suppose* $\mathbf{f}$ *is a homeomorphism. Let* $(\mathbf{s}^*_\mathbf{f}, \mathbf{f}^*)$ *be a second projection index-surface pair with* $\mathbf{f}(\mathbf{s_f}(\mathbf{x})) = \mathbf{f}^*(\mathbf{s}_{\mathbf{f}^*}(\mathbf{x}))$ *for all* $\mathbf{x}$ *and* $\mathbf{f}^*$ *a homeomorphism. Then there exists homeomorphism* ψ *with*

$$\mathbf{s}_{\mathbf{f}^*}(\mathbf{x}) = \psi(\mathbf{s_f}(\mathbf{x}))$$

for all $\mathbf{x}$.

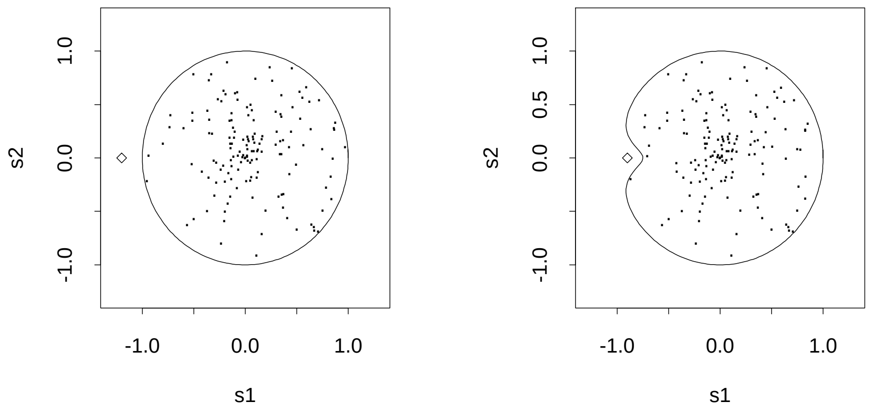
The result makes several assumptions that may seem restrictive. The assumption that curve or surface $\mathbf{f}$ is a homeomorphism is equivalent to assuming the curve or surface does not intersect itself, e.g., there are no "loops." The assumption that $\mathbf{f}(\mathbf{s_f}(\mathbf{x})) = \mathbf{f}^*(\mathbf{s}_{\mathbf{f}^*}(\mathbf{x}))$ for all $\mathbf{x}$ has two implications: (1) that $\mathbf{f}$ and $\mathbf{f}^*$ trace the same curve; and (2) that the respective projection indices map ambiguity points to the same points on the curve. Thus the result does not apply to curves distinct from $\mathbf{f}$ that also satisfy the least-squares objective function, e.g., problems where there are multiple minima to the objective function.

The implications of this theorem are different for curves and surfaces and will be discussed separately. In the case of curves, the score values are a vector and the *order* of the elements of a vector is preserved under homeomorphic transformations, i.e., if $\mathbf{s}$ is a score vector and ψ is a homeomorphic transformation, then $s_i < s_j$ implies either $\psi(s_i) < \psi(s_j)$ or $\psi(s_i) > \psi(s_j)$. As a result, score values must be treated as *ordinal* data and cannot be treated as continuous data, i.e., the magnitude of a difference between two score values is not interpretable but their order is. Note that this level of identification is sufficient for the control-chart applications mentioned above.

Surfaces are much more problematic because the orientation of a set of points can be greatly distorted by a homeomorphic transformation. Basic properties like convexity are not preserved under such transformations.

[10]Function $\mathbf{f}$ is a homeomorphism if and only if it is continuous and has a continuous inverse.

Consider the following hypothetical quality control example, which is similar to the application discussed in [5]. Suppose that a large number of process variable (p large) are monitored and that the p-dimensional observations lie on a 2-dimensional surface, i.e., the intrinsic dimension of the process variables is $r = 2$. Plotting the 2-dimensional score vectors on a plane as shown in Fig. 9a reveals that points where the process is "in control" (solid points) lie within a circular region while an "out-of-control" point (plotted as $\diamond$) lies outside the region. One might establish the following procedure: the process is judged to be "in control" when the score value falls within the circular region and "out of control" otherwise. The problem with this procedure is that the circular "in-control" region resulted from the choice of starting values and training method. If the surface had been extracted with different starting values or a different training method, the parameterization of the surface and thus the score values would most likely be different, possibly yielding an "in-control" region with a shape other than circular.



(a) Circular "in-control" region

(b) Heart-shaped "in-control" region

Figure 9: Effect of a homeomorphic transformation on "in-control" region

I now show how the shape of the in-control region could be distorted. Suppose that the surface extracted by the simultaneous method is parameterized by polar coordinates, (r, θ). Theorem 1 guarantees that the score values from any other solution can be related to the current score values with a homeomorphic (continuous and one-to-one) transformation. Consider the homeomorphic transformation

$$\psi \begin{pmatrix} r \\ \theta \end{pmatrix} = \begin{pmatrix} r \left(1 - \frac{\exp -10(\pi-\theta)}{(1+\exp -10(\pi-\theta))^2}\right) \\ \theta \end{pmatrix},$$

which transforms the circular boundary to the heart-shaped boundary shown in Fig. 9b. The expression $\exp(-x)/(1+\exp(-x))^2$ is the derivative of the logistic function and has a shape similar to the Gaussian distribution. Points near $\theta = \pi$ are moved closer to the origin while the locations of all other points are nearly unchanged. The transformation is continuous and one-to-one. After the transformation, it is not so clear that the out-of-control point ($\diamond$) lies outside the normal operating range.

5 Conclusions

NLPCA estimates a curve or surface $\mathbf{f}$ passing through the middle of the input data. The first three layers of an NLPCA network, the projection index $s_{\mathbf{f}}$ projects the inputs onto $\mathbf{f}$ and the last three layers define $\mathbf{f}$. NLPCA models the projection index with a continuous function, which has several implications. Projections onto (nonlinear) curves or surfaces are suboptimal resulting in larger training value error than methods with discontinuous projection indices like principal curves. NLPCA cannot approximate curves or surfaces that intersect themselves and it cannot parameterize a curve or surface with a parameterization that has discontinuous jumps. Any two parameterizations of a surface can be related by a homeomorphic transformation. Under this level of identification, the order of the score variable values for curves are preserved, but the score values for surfaces can change greatly, possibly leading to different interpretations.

An important area of future research for applications where scores are interpreted is defining and finding "useful" parameterizations for surfaces. This question has been examined thoroughly in the case of PCA and factor analysis. After deciding on the dimension of the subspace for a given problem, i.e., picking r, and extracting a set of basis vectors spanning the subspace, researchers in the social sciences often select an alternative set of basis vectors (the parameterization) for the subspace that has certain properties and is more interpretable than the original basis vectors resulting from the PCA or factor analysis estimation. This change of basis is called a *rotation* in the social science literature. See [13, § 9.6] or [2, § 3.4] for further discussion. Before the surface-score values can be interpreted, similar work must be done for the nonlinear feature extraction methods.

Proof of Result 1

Denote the range of $\mathbf{s_f}$ by $S = \{\mathbf{s} : \mathbf{s} = \mathbf{s_f}(\mathbf{x}), \mathbf{x} \in \mathcal{R}^p\}$, the range of $\mathbf{s}^*_\mathbf{f}$ by $S^* = \{\mathbf{s} : \mathbf{s} = \mathbf{s}^*_\mathbf{f}(\mathbf{x}), \mathbf{x} \in \mathcal{R}^p\}$, and the inverse image of $\mathbf{s} \in S$ by $\mathbf{s_f}^{-1}(\mathbf{s}) = \{\mathbf{x} : \mathbf{s_f}(\mathbf{x}) = \mathbf{s}\}$.

To show that ψ is a homeomorphism, one must show that ψ is one-to-one and continuous. We first show that ψ is one-to-one. Suppose ψ is not one-to-one. Then either (1) function ψ exists but is not one-to-one, or (2) function ψ does not exist.

First assume (1). Since ψ is a function, for all $\mathbf{s} \in S$ there is a unique $\mathbf{s}^* \in S^*$ with $\mathbf{s}^* = \psi(\mathbf{s})$; since $\mathbf{f}(\mathbf{s_f}(\mathbf{x})) = \mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}))$ and $\mathbf{f}$ and $\mathbf{f}^*$ are 1-1, for all $\mathbf{x} \in \mathbf{s_f}^{-1}(\mathbf{s})$, $\mathbf{s}^*_\mathbf{f}(\mathbf{x}) = \psi(\mathbf{s_f}(\mathbf{x}))$. Since ψ is not one-to-one there exist $\mathbf{s}_1, \mathbf{s}_2 \in S$ with $\psi(\mathbf{s}_1) = \psi(\mathbf{s}_2) = \mathbf{s}^*$, for some $\mathbf{s}^* \in S^*$. Let $\mathbf{x}_1 \in \mathbf{s_f}^{-1}(\mathbf{s}_1)$ and $\mathbf{x}_2 \in \mathbf{s_f}^{-1}(\mathbf{s}_2)$. Since $\mathbf{f}$ is 1-1, $\mathbf{f}(\mathbf{s}_1) \neq \mathbf{f}(\mathbf{s}_2)$. But,

$$
\begin{aligned}
\mathbf{f}(\mathbf{s}_1) &= \mathbf{f}(\mathbf{s_f}(\mathbf{x}_1)) \\
&= \mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}_1)) \\
&= \mathbf{f}^*(\mathbf{s}^*) \\
&= \mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}_2)) \\
&= \mathbf{f}(\mathbf{s_f}(\mathbf{x}_2)) \\
&= \mathbf{f}(\mathbf{s}_2) \qquad (15)
\end{aligned}
$$

This is a contradiction. Therefore the statement must be true in case (1).

Now assume (2). Since function ψ does not exist, there exist $\mathbf{x}_1$ and $\mathbf{x}_2$ with $\mathbf{s_f}(\mathbf{x}_1) \neq \mathbf{s_f}(\mathbf{x}_2)$ but $\mathbf{s}^*_\mathbf{f}(\mathbf{x}_1) = \mathbf{s}^*_\mathbf{f}(\mathbf{x}_2)$. Since $\mathbf{s_f}$ and $\mathbf{s}^*_\mathbf{f}$ are 1-1, $\mathbf{f}(\mathbf{s_f}(\mathbf{x}_1)) \neq \mathbf{f}(\mathbf{s_f}(\mathbf{x}_2))$ and $\mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}_1)) = \mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}_2))$. But,

$$
\begin{aligned}
\mathbf{f}(\mathbf{s_f}(\mathbf{x}_1)) &= \mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}_1)) \\
&= \mathbf{f}^*(\mathbf{s}^*_\mathbf{f}(\mathbf{x}_2)) \\
&= \mathbf{f}(\mathbf{s_f}(\mathbf{x}_2)). \qquad (16)
\end{aligned}
$$

This is also a contradiction. Therefore ψ is one-to-one. Proving that ψ is continuous is straightforward and follows from the definition of continuous.

Acknowledgments

I thank Ajit Tamhane, Richard Mah, Tom Severini, and Rick Briesch for many helpful discussions. The computing equipment used for the empirical results presented in this paper was partially funded by NSF Grant DMS-9505799.

References

[1] B. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.

[2] D. J. Bartholomew, *Latent Variable Models and Factor Analysis*, Charles Griffin & Company LTD, London, 1987.

[3] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Systems*, vol. 2, pp. 303–314, 1989.

[4] David DeMers and Garrison Cottrell, "Nonlinear dimensionality reduction," *Neural Information processing systems*, vol. 5, pp. 580–587, 1993.

[5] D. Dong and T.J. McAvoy, "Nonlinear principal component analysis — based on principal curves and neural networks," *Computers and Chemical Engineering*, vol. 20, no. 1, pp. 65–78, 1996.

[6] J.H. Friedman, "Exploratory projection pursuit," *JASA*, vol. 82, no. 397, pp. 249–267, 1987.

[7] Colin Fyfe and Roland Badderey, "Non-linear data structure extraction using simple hebbian networks," Tech. Rep., Department of Computer Science, University of Strathclyde, 1995.

[8] T. Hastie, *Principal Curves and Surfaces*, Ph.D. thesis, Stanford University, November 1984.

[9] T. Hastie and W. Stuetzle, "Principal curves," *JASA*, vol. 84, no. 406, pp. 502–516, 1989.

[10] Juha Karhunen and Jyrki Joutsensalo, "Representation and separation of singnals using nonlinear pca type learning," *Neural Networks*, pp. 113–127, 1994.

[11] M.A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[12] M. LeBlanc and R. Tibshirani, "Adaptive principal surfaces," *JASA*, vol. 89, no. 425, pp. 53–64, 1994.

[13] K.V. Mardia, J.T. Kent, and J.M. Bibby, *Multivariate Analysis*, Academic Press, London, 1979.

[14] D.E. Rumelhart, G.E. Hinton, and J.L. McClelland, "Learning internal representations by error propagation," in *Parallel Distributed Processing Volume 1: Foundations*, pp. 318–362. MIT Press, 1986.

[15] Terence D. Sanger, "Optimal hidden units for two-layer nonlinear feedforward neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, no. 4, pp. 545–561, 1991.

[16] D.J. Struik, *Lectures on Classical Differential Geometry*, Dover Publication, Inc., New York, second edition, 1961.

[17] Shufeng Tan and Michael Mavrovouniotis, "Reducing data dimensionality through optimizing neural network inputs," *AIChE Journal*, 1995, in press.

[18] J.A. Thorpe, *Elementary Topics in Differential Geometry*, Springer-Verlag, New York, 1979.