Chapter 3

Principal Curves and Related Areas

In this chapter we introduce the original HS definition of principal curves and summarize some of the subsequent research. We also describe the connection of principal curves to some of the related unsupervised learning models. Section 3.1 introduces the HS definition of principal curves, and describes the HS algorithm for probability distributions and data sets. Section 3.2 summarizes some of the subsequent results on principal curves and highlights the relationship between principal curves, self-organizing maps and nonlinear principal component analysis.

3.1 Principal Curves with Self-Consistency Property

3.1.1 The HS Definition

Property 3 in Section 2.2 states that for elliptical distributions the first principal component is self-consistent, i.e., any point of the line is the conditional expectation of $\mathbf{X}$ over those points of the space which project to this point. HS generalized the self-consistency property of principal components and defined principal curves as follows.

Definition 2 *The smooth curve* $\mathbf{f}(t)$ *is a principal curve if the following hold:*

(i) $\mathbf{f}$ *does not intersect itself,*

(ii) $\mathbf{f}$ *has finite length inside any bounded subset of* $\mathbb{R}^d$*,*

(iii) $\mathbf{f}$ *is self-consistent, i.e.,* $\mathbf{f}(t) = E\left[\mathbf{X}|t_{\mathbf{f}}(\mathbf{X}) = t\right]$.

Intuitively, self-consistency means that each point of $\mathbf{f}$ is the average (under the distribution of $\mathbf{X}$) of all points that project there. Thus, principal curves are smooth self-consistent curves which pass through the "middle" of the distribution and provide a good one-dimensional nonlinear summary of the data (see Figure 6).

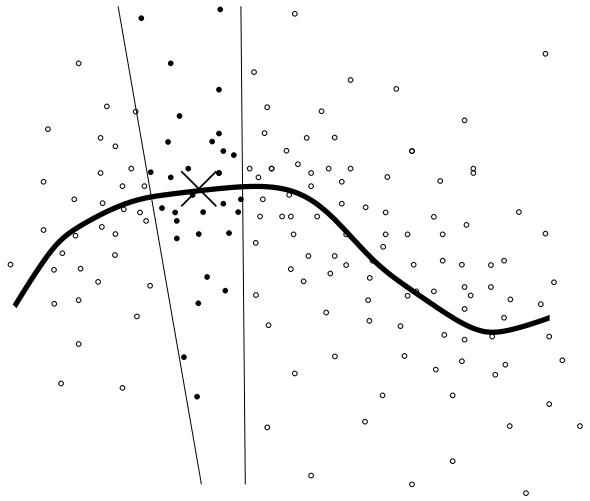

Figure 6: Self-consistency. Each point of the curve is the average of points that project there.

It follows from the discussion in Section 2.2 that the principal component lines are stationary points of the distance function. HS proved an analogous result for principal curves. Formally, let $\mathbf{f}$ be a smooth (infinitely differentiable) curve, and for $\lambda \in \mathbb{R}$ consider the perturbation $\mathbf{f}+\lambda\mathbf{g}$ of $\mathbf{f}$ by a smooth curve $\mathbf{g}$ such that $\sup_t \|\mathbf{g}(t)\| \leq 1$ and $\sup_t \|\mathbf{g}'(t)\| \leq 1$. Then $\mathbf{f}$ is a principal curve if and only if $\mathbf{f}$ is a stationary point of the distance function in the sense that for all such $\mathbf{g}$,

$$\left.\frac{\partial \Delta(\mathbf{f}+\lambda\mathbf{g})}{\partial \lambda}\right|_{\lambda=0} = 0.$$

In this sense the HS principal curve definition is a natural generalization of principal components.

Existence of the HS Principal Curves

The existence of principal curves defined by the self-consistency property is in general an open question. Until recently, the existence of principal curves had been proven only for some special distributions, such as elliptical or spherical distributions, or distributions concentrated on a smooth curve. The first results on principal curves of non-trivial distributions are due to Duchamp and Stuetzle [DS96a] who studied principal curves in the plane. They showed that principal curves are solutions of a differential equation. By solving this differential equation for uniform densities on rectangles and annuli, they found oscillating principal curves besides the obvious straight and circular ones, indicating that principal curves in general will not be unique. They also showed that if a density has several principal curves, they have to cross, a property somewhat analogous to the orthogonality of principal components.

The HS Algorithm for Distributions

Based on the self-consistency property, HS developed an algorithm for constructing principal curves. Similar in spirit to the GL algorithm of vector quantizer design (Section 2.1), and the RTB algorithm (Section 2.2.4) of principal component analysis, the HS algorithm iterates between a projection step and an expectation step until convergence. In the projection step, projection indices of the data to the curve are computed. In the expectation step, a new curve is computed. For every point $\mathbf{f}^{(j)}(t)$ of the previous curve, a point of the new curve is defined as the expectation of the data that project to $\mathbf{f}^{(j)}(t)$. When the probability density of $\mathbf{X}$ is known, the formal algorithm for constructing principal curves is the following.

Algorithm 4 (The HS algorithm)

Step 0 *Let* $\mathbf{f}^{(0)}(t)$ *be the first principal component line for* $\mathbf{X}$. *Set* $j = 0$.

Step 1 (Projection) *Set* $t_{\mathbf{f}^{(j)}}(\mathbf{x}) = \max\{t : \|\mathbf{x} - \mathbf{f}(t)\| = \min_{\tau} \|\mathbf{x} - \mathbf{f}(\tau)\|\}$ *for all* $\mathbf{x} \in \mathbb{R}^d$.

Step 2 (Expectation) *Define* $\mathbf{f}^{(j+1)}(t) = E\left[\mathbf{X} | t_{\mathbf{f}^{(j)}}(\mathbf{X}) = t\right]$.

Step 3 *Stop if* $\left(1 - \frac{\Delta(\mathbf{f}^{(j+1)})}{\Delta(\mathbf{f}^{(j)})}\right)$ *is less than a certain threshold. Otherwise, let* $j = j+1$ *and go to Step 1.*

Although HS is unable to prove that the algorithm converges, they have the following evidence in its favor:

1. By definition, principal curves are fixed points of the algorithm.
2. Assuming that each iteration is well defined and produces a differentiable curve, the expected squared distance $\Delta(\mathbf{f})$ converges.
3. If Step 1 is replaced by fitting a least squares straight line, then the procedure converges to the largest principal component.

Unfortunately, the fact that $\Delta(\mathbf{f})$ converges does not mean that $\mathbf{f}$ converges to any meaningful solution. Among the principal components, the largest principal component minimizes the distance function, the smallest principal component maximizes it, and all the others are saddle points. Interestingly, there is no such distinction between different principal curves of a distribution. [DS96b] showed that all principal curves are saddle points of the distance function. In this sense, any algorithm that aims to find a principal curve by minimizing the distance function will fail to converge to a stable solution without further restricting the set of admissible curves. This fact is one of the motivations behind our new definition of principal curves in Chapter 4.

3.1.2 The HS Algorithm for Data Sets

Similarly to the GL and RTB algorithms, the HS algorithm can be extended to data sets. Unlike in the former two cases, however, this case requires more than simple replacements of expectations by the corresponding sample averages. A general issue is the representation of the curve by a finite number of parameters. A more serious problem arises in the expectation step: in general there is at most one point that projects to a given point of the curve. In this section we give a detailed treatment of the modifications proposed by HS, and analyze the algorithm.

Assume that a set of points $\mathcal{X}_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ is given. Project the data points to an arbitrary curve $\mathbf{f}$, and index the points so that their projection indices $t_1, \ldots, t_n$ are in increasing order. We can represent the curve $\mathbf{f}$ by a polygonal curve of n vertices by connecting pairs of consecutive projection points $(\mathbf{f}(t_i), \mathbf{f}(t_{i+1})), i = 1, \ldots, n-1$ by line segments. In the discussion below we assume that all curves produced by the algorithm are such polygonal curves. We also assume that all curves are arc length parameterized, so the parameters $t_i, i = 1, \ldots, n$ can be defined recursively by

$$\begin{aligned} &1. \quad t_1 = 0, \\ &2. \quad t_i = t_{i-1} + \|\mathbf{f}(t_i) - \mathbf{f}(t_{i-1})\|, \;\; i = 2, \ldots, n. \end{aligned} \tag{38}$$

In Step 0, $\mathbf{f}^{(0)}(t)$ is the first principal component line of the data set $\mathcal{X}_n$. In the stopping condition in Step 3, the distance function $\Delta\left(\mathbf{f}^{(j+1)}\right)$ is replaced by the empirical distance function,

$$\Delta_n\left(\mathbf{f}^{(j+1)}\right) = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \mathbf{f}^{(j+1)}\left(t_i^{(j)}\right)\right\|^2.$$

In the projection step (Step 1), the new projection indices $t_i^{(j)}, i = 1, \ldots, n$ are computed by projecting the data points to $\mathbf{f}^{(j)}(t)$. When we reach this step for the first time, the projection indices can be set by projecting the data points to the first principal component line. After the jth iteration, the current curve is represented as a polygonal curve of vertices $\mathbf{f}^{(j)}\left(t_1^{(j-1)}\right), \ldots, \mathbf{f}^{(j)}\left(t_n^{(j-1)}\right)$. Let $\mathbf{s}_\ell$ be the line segment that connects the vertices $\mathbf{f}^{(j)}\left(t_\ell^{(j-1)}\right)$ and $\mathbf{f}^{(j)}\left(t_{\ell+1}^{(j-1)}\right)$, $\ell = 1, \ldots, n-1$. To compute the projection index of a data point $\mathbf{x}_i$, we have to find the nearest line segment to $\mathbf{x}_i$, denoted by $\mathbf{s}_{\ell(i)}$, where the index $\ell(i)$ is defined by

$$\ell(i) = \underset{\ell=1,\ldots,n-1}{\arg\min}\; \Delta(\mathbf{x}_i, \mathbf{s}_\ell).$$

If $\mathbf{s}_{\ell(i)}$ is defined over $\left[t_{\ell(i)}^{(j-1)}, t_{\ell(i)+1}^{(j-1)}\right]$, the new projection index is identical to the projection index of $\mathbf{x}_i$ to $\mathbf{s}_{\ell(i)}$, that is,

$$t_i^{(j)} = t_{\mathbf{s}_{\ell(i)}}(\mathbf{x}_i).$$

Figure 7 illustrates the method for a data point.

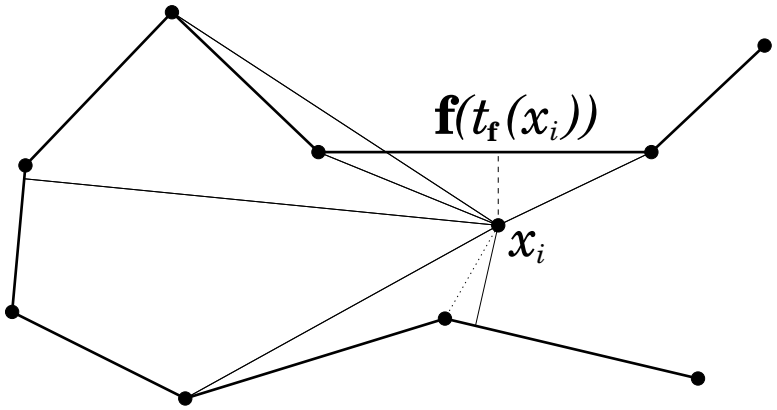

Figure 7: Computing projection points. $\mathbf{x}_i$ is projected to the line segments (along the solid thin lines), and the nearest projection point is chosen (connected to $\mathbf{x}_i$ by dashed line). Note that the nearest vertex (connected to $\mathbf{x}_i$ by dotted line) is not the endpoint of the nearest line segment.

There seems to be a simpler approach to find the projection point of a data point $\mathbf{x}_i$. Instead of searching through the line segments, one could find the nearest vertex to $\mathbf{x}_i$, project the point to the vertex and the two incident line segments, and pick the nearest projection. Figure 7 clearly indicates that this approach can yield a wrong result if the nearest vertex of the polygonal curve is not the endpoint of nearest line segment. Although this configuration occurs quite rarely, in general it cannot be excluded.

Before proceeding with expectation step, the data points are reindexed in increasing order by their projection indices.

In the expectation step (Step 2), finite points of the new curve $\mathbf{f}^{(j+1)}(t) = E\left[\mathbf{X}|t_{\mathbf{f}^{(j)}}(\mathbf{X}) = t\right]$ are estimated at the n projection indices $t = t_1^{(j)}, \ldots, t_n^{(j)}$ found in the projection step. In general, the only observation that projects to $\mathbf{f}^{(j)}(t)$ at $t_i^{(j)}$ is $\mathbf{x}_i$. Using this one point in the averaging would result in a curve that visits all the data points after the first iteration. To tackle this problem, HS proposed two approaches. In the first, $E\left[\mathbf{X}|t_{\mathbf{f}^{(j)}}(\mathbf{X}) = t_i^{(j)}\right]$ is estimated by averaging over observations that *project close* to $t_i^{(j)}$. HS used the *locally weighted running-lines smoother* [Cle79] for local averaging. In the second approach, a non-parametric regression estimate (*cubic smoothing splines* [Sil85]) is used to minimize a data-dependent criteria.

Locally Weighted Running-Lines Smoother

Consider the estimation of the single coordinate function $E\left[X|t_{\mathbf{f}^{(j)}}(X) = t_i^{(j)}\right]$ based on the sample of n pairs $(t_1^{(j)}, x_1), \ldots, (t_n^{(j)}, x_n)$. To estimate this quantity, the smoother fits a straight line to the first wn observations $\{x_k\}$ of which the projection index $t_k^{(j)}$ is the closest to $t_i^{(j)}$. The estimate is taken to be the fitted value of the line at $t_i^{(j)}$. The fraction w of points in the neighborhood is called the *span*, and w is a parameter of the smoother. In fitting the line, weighted least squares regression is used.

The weights are derived from a symmetric kernel centered at $t_i^{(j)}$ that goes smoothly to 0 within the neighborhood. Formally, let $t_w^{(j)}$ denote the wnth nearest projection index to $t_i^{(j)}$, and define the weight w_{ik} of the observation x_k by

$$w_{ik} = \begin{cases} \left(1 - \left|\frac{t_k^{(j)} - t_i^{(j)}}{t_w^{(j)} - t_i^{(j)}}\right|^3\right)^{1/3} & \text{if } \left|t_k^{(j)} - t_i^{(j)}\right| < \left|t_w^{(j)} - t_i^{(j)}\right|, \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

Cubic Smoothing Splines

The algorithm to estimate principal curves for data sets is motivated by the algorithm for finding principal curves of distributions, so it is designed to find a stationary point of the average squared distance, $\Delta_n(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^n \|\mathbf{x}_i - \mathbf{f}(t_\mathbf{f}(\mathbf{x}_i))\|^2$. To obtain a smooth curve solution, motivated by cubic smoothing splines [Sil85], HS suggested to minimize a penalized average squared distance criterion to define principal curves. Formally, let

$$G(\mathbf{f}) = \Delta_n(\mathbf{f}) + \mu P(\mathbf{f}), \tag{40}$$

where $P(\mathbf{f}) = \int_0^1 \|\mathbf{f}''(\tau)\|^2 d\tau$ measures the total curvature of the curve, and the penalty coefficient μ is a parameter of the algorithm. Note that the parameter of the curve is rescaled to lie in the interval $[0,1]$. In the expectation step the criteria (40) is minimized by minimizing separately the d coordinate functions,

$$G(f_\ell) = \frac{1}{n}\sum_{i=1}^n |x_{i\ell} - f_\ell(t_i)|^2 + \mu \int_0^1 |f_\ell''(\tau)|^2 d\tau, \quad \ell = 1, \ldots, d.$$

Computational Complexity of the HS Algorithm

In the projection step the distance between n line segments and n data points is computed, so the complexity of the step is $O(n^2)$. The computational complexity of the expectation step is $O(n^2)$ for the kernel type smoothers, and $O(n)$ for the smoothing spline. The complexity of the sorting routine after the projection step is $O(n \log n)$. Hence, the computational complexity of the HS algorithm, dominated by the complexity of the projection step, is $O(n^2)$.[1]

3.1.3 The Bias of the HS Algorithm

HS observed two sources of bias in the estimation process. *Model bias* occurs when data points are generated by the additive model

$$\mathbf{X} = \mathbf{f}(Y) + \mathbf{e} \tag{41}$$

[1] [Has84] argues that the computational complexity of the projection step can be improved by using spatial data structures. [YMMS92] claims similar results. However, both [Has84] and [YMMS92] attempt to find the projection point $\mathbf{x}$ of a data point by finding the nearest *vertex* to $\mathbf{x}$, and projecting $\mathbf{x}$ to the two line segments incident to the vertex. As we showed earlier in this section, in general, this approach can yield a wrong result.

where Y is uniformly distributed over the domain of the smooth curve $\mathbf{f}$, and $\mathbf{e}$ is bivariate additive noise which is independent of Y. In general, if $\mathbf{f}$ has a curvature, it is not self-consistent so it is not a principal curve of the distribution of $\mathbf{X}$. The self-consistent curve lies *outside* $\mathbf{f}$ from the point of view of the center of the curvature. This bias goes to 0 with the ratio of the noise variance and the radius of the curvature.

Estimation bias occurs because the scatterplot smoothers average over neighborhoods. The estimation bias points towards the center of curvature so usually it has a flattening effect on the estimated curve. Unlike the model bias, the estimation bias can be affected by parameters of the algorithm. The larger the span coefficient w of the running-lines smoother or the penalty coefficient μ of the spline smoother, the larger is the bias. So, in theory it is possible to set these parameters so that the two bias components cancel each other.

HS proposed a simple model for the quantitative analysis of the two bias components. Let $\mathbf{f}$ be an arc length parameterized circle with constant curvature $1/r$, i.e, let

$$\mathbf{f}(t) = \begin{bmatrix} r\cos(t/r) \\ r\sin(t/r) \end{bmatrix}$$

for $t \in I = [-r\pi, r\pi)$. Let the random variable $\mathbf{X}$ be defined by (41). Assume that the noise $\mathbf{e}$ has zero mean and σ^2 variance in both coordinates. HS showed that in this situation the radius r^* of the self-consistent circle $\mathbf{f}^*$ is larger than r. The intuitive explanation of this is that the model (41) seems to generate more mass outside the circle $\mathbf{f}$ than inside (Figure 8(a)). In a quantitative analysis, HS showed that, under certain conditions,

$$r^* \approx r + \frac{\sigma^2}{2r} \tag{42}$$

so the bias inherent in the model (41) is $\sigma^2/2r$. It also follows from the analysis that the distance function at $\mathbf{f}^*$ is

$$\Delta(\mathbf{f}^*) \approx \sigma^2 - \frac{\sigma^4}{4r^2} = \Delta(\mathbf{f}) - \frac{\sigma^4}{4r^2}. \tag{43}$$

The source of the estimation bias is the local averaging procedure used in the HS algorithm designed for data sets. Assume that the principal curve at $t = 0$ is estimated by using data that projects to the curve in the interval $I_\theta = [-r\theta, r\theta]$ (Figure 8(b)). The smoother fits a straight line to the data, and the estimate is taken to be the fitted value of the line at $t = 0$. Clearly, the estimate will be *inside* the generating curve. HS showed that under certain conditions the radius of the estimated curve is

$$r_\theta = r^* \frac{\sin(\theta/2)}{\theta/2}. \tag{44}$$

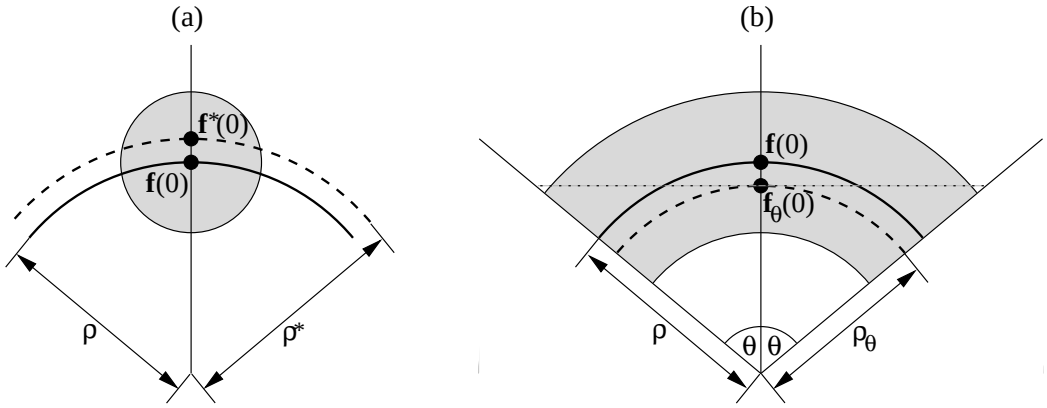

Figure 8: (a) The model bias. There is more mass outside **f** than inside so the self-consistent circle has a larger radius than the generating curve. (b) The estimation bias. A straight line (dotted line) is fitted to the data. The estimated point $\mathbf{f}_\theta(0)$ is *inside* the generating circle.

Reducing the Estimation Bias

It follows from (42) and (44) that the span θ can be chosen so that the estimation bias and the model bias are approximately balanced. Unfortunately, for moderate sample sizes, the obtained span tends to be too small. In other words, if the span size is set to an appropriate value for a *given sample size*, the estimation bias tends to be much larger than the model bias. This observation naturally created a demand for procedures to reduce the estimation bias.

Banfield and Raftery [BR92] (hereafter BR) proposed the following modifications to the algorithm. The expectation step (Step 3) in the HS algorithm can be rewritten as

$$\mathbf{f}^{(j+1)}(t) = \mathbf{f}^{(j)}(t) + \mathbf{b}^{(j)}(t)$$

where

$$\mathbf{b}^{(j)}(t) = E\left(\mathbf{X} - \mathbf{f}^{(j+1)}(t) \middle| t_{\mathbf{f}^{(j)}}(\mathbf{X}) = t\right)$$

can be considered as a measure of the bias of $\mathbf{f}^{(j+1)}$ at t. Let

$$\mathbf{p}_i^{(j)} = \mathbf{x}_i - \mathbf{f}^{(j)}\left(t_i^{(j)}\right)$$

denote the *projection residual* of the data point $\mathbf{x}_i$ projected onto $\mathbf{f}^{(j)}$. The bias measure $\mathbf{b}^{(j)}(t)$ is the expected value of the projection residuals of the data points that project onto $\mathbf{f}^{(j)}$ at t. [BR92] suggested that, in the algorithm for data sets, the projection residuals of the data points in $\mathcal{X}$, rather then the data points themselves, should be used to calculate $\mathbf{f}^{(j+1)}(t)$. Accordingly, let

$$\overline{\mathbf{p}}_i^{(j)} = \frac{\sum_{k=1}^n w_{ik}\mathbf{p}_k^{(j)}}{\sum_{k=1}^n w_{ik}}$$

be the weighted average of the projection residuals of the data points that project close to $t_i^{(j)}$. By using $\overline{\mathbf{p}}_i^{(j)}$ as the estimation of the bias $\mathbf{b}^{(j)}\left(t_i^{(j)}\right)$, the new point of the curve is estimated by

$$\mathbf{f}^{(j+1)}\left(t_i^{(j)}\right) = \mathbf{f}^{(j)}\left(t_i^{(j)}\right) + \overline{\mathbf{p}}_i^{(j)}.$$

[BR92] also extended the HS algorithm to closed curves. Experimental results on simulated data are given in Section 5.2, where the HS algorithm with smoothing splines is compared to the BR algorithm and the polygonal line algorithm introduced in Section 5.1.

3.2 Alternative Definitions and Related Concepts

3.2.1 Alternative Definitions of Principal Curves

Two substantially different approaches to principal curves have been proposed subsequent to Hastie and Stuetzle's groundbreaking work. Tibshirani [Tib92] introduced a semi-parametric model for principal curves. The motivation of [Tib92] to redefine principal curves is the unsettling property of the HS principal curves that if the distribution of the data is defined by the additive model $\mathbf{X} = \mathbf{f}(Y) + \mathbf{e}$ (see (41)), $\mathbf{f}$ is not the principal curve of $\mathbf{X}$ in general. To solve this problem, [Tib92] derives principal curves from the additive model (41). Consider a d-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_d)$ with density $\mu_{\mathbf{X}}$. Now imagine that $\mathbf{X}$ was generated in two stages. In the first step, a point on a curve $\mathbf{f}(Y)$ is generated according to some distribution μ_Y, and in the second step, $\mathbf{X}$ is generated from a conditional distribution $\mu_{\mathbf{X}|Y}$ where the mean of $\mu_{\mathbf{X}|Y}$ is $\mathbf{f}(Y)$, and $X_1, \ldots, X_d$ are conditionally independent given Y. Using this model, [Tib92] defines principal curves as follows.

Definition 3 *The* principal curve *of a random variable* $\mathbf{X}$ *is a triplet* $\{\mu_Y, \mu_{\mathbf{X}|Y}, \mathbf{f}\}$ *satisfying the following conditions:*

(i) μ_Y *and* $\mu_{\mathbf{X}|Y}$ *are consistent with* $\mu_{\mathbf{X}}$*, that is,* $\mu_{\mathbf{X}}(\mathbf{x}) = \int \mu_{\mathbf{X}|Y}(\mathbf{x}|y)\mu_Y(y)dy$.

(ii) $X_1, \ldots, X_d$ *are conditionally independent given* Y.

(iii) $\mathbf{f}(t)$ *is a curve in* $\mathbb{R}^d$ *parameterized over a closed interval in* $\mathbb{R}$ *satisfying* $\mathbf{f}(t) = E[\mathbf{X}|Y = t]$.

It is easy to see that if the distribution of the data is defined by the additive model (41), the generating curve $\mathbf{f}$ is indeed the principal curve of $\mathbf{X}$ in the sense of Definition 3. Based on this definition, [Tib92] proposed a semi-parametric scheme for estimating principal curves of data sets. In the model, μ_Y is left completely unspecified, while $\mu_{\mathbf{X}|Y}$ is assumed to be from a parametric family. Therefore, at a certain parameter y, one has to estimate the point of the curve $\mathbf{f}(y)$ and the

parameters $\Sigma(y)$ of $\mu_{\mathbf{X}|Y}$. Given a data set $\mathcal{X}_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$, [Tib92] uses maximum likelihood estimation to find the unknown parameters. The log-likelihood

$$l(\mathbf{f}, \Sigma) = \sum_{i=1}^{n} \log \int \mu_{\mathbf{X}|Y}(\mathbf{x}_i|\mathbf{f}(y), \Sigma(y))\mu_Y(y)dy$$

was minimized by using the EM algorithm [DLR77]. The algorithm was tested on several simulated and real data sets and compared to the HS algorithm. Although Definition 3 has some theoretical advantages over the HS definition, the resulting estimation procedure does not produce better results than the HS algorithm.

Recently, Delicado [Del98] proposed yet another definition based on a property of the first principal components of multivariate normal distributions. Consider a d-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_d)$. [Del98] calls a point $\mathbf{x}^* \in R^d$ a *principal oriented point* if there exists a direction $\mathbf{u}^*(\mathbf{x}^*) \in R^d$ such that $\mathbf{x}^*$ is the conditional mean of $\mathbf{X}$ in the hyperplane orthogonal to $\mathbf{u}^*(\mathbf{x}^*)$ that contains $\mathbf{x}^*$, i.e.,

$$\mathbf{x}^* = E\left[\mathbf{X}|(\mathbf{X} - \mathbf{x}^*)^T \mathbf{u}^*(\mathbf{x}) = 0\right].$$

A curve $\mathbf{f} : [a, b] \to R^d$ is called a *principal curve of oriented points* of $\mathbf{X}$ if for all $t \in [a, b]$, the points of the curve $\mathbf{f}(t)$ are principal oriented points. The definition can be considered as a generalization of PCA since the first principal component of a multivariate normal distribution is a principal curve of oriented points. It can be seen easily that if the curve $\mathbf{f}$ satisfies certain regularity conditions, namely that no points of the support of the distribution of $\mathbf{X}$ can be projected orthogonally to more than one points of $\mathbf{f}$, then $\mathbf{f}$ is a principal curve of oriented points if and only if it is a principal curve in the HS sense. [Del98] also proposed an algorithm to find a principal curve of oriented points of a given data set. Examples indicate that the curves produced by the procedure tend to be less smooth than the curves produced by the HS algorithm.

3.2.2 The Self-Organizing Map

Kohonen's self-organizing map (SOM) [Koh82] is one of the most widely used and most extensively studied unsupervised learning method. The basic idea of the algorithm was inspired by the way the brain forms topology preserving neural representations or maps of various sensory impressions. Keys of the success of the SOM among practitioners are its simplicity, efficiency, and low computational complexity.

In a certain sense, the SOM algorithm can be considered as a generalization of both the GL algorithm and the HS algorithm (with local averaging). The relation of SOM and vector quantization is a widely known fact (see, e.g. [Koh97]), whereas the similarities between SOM and principal curves were pointed out recently by [RMS92] and [MC95]. The SOM algorithm is usually formulated as a stochastic learning algorithm. To emphasize its similarities to the HS and GL algorithms,

we present it here as a batch method as it was first formulated by Luttrell [Lut90].

In its original form, the SOM is a nearest neighbor vector quantizer equipped with a topology. Similarly to vector quantization, we are given a set of codepoints $\mathcal{C} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subset \mathbb{R}^d$. In addition, there is a weighted graph defined over $\mathcal{C}$ by a $k \times k$ matrix of weights $\mathbf{W} = \{w_{\ell,m}\}$. The weights $w_{\ell,m}$ $(\ell, m = 1, \ldots, k)$ are usually defined as a monotonically decreasing function of the Euclidean distance between the initial codepoints $\mathbf{v}_\ell$ and $\mathbf{v}_m$. In this sense, $\mathbf{W}$ can be considered as a topology over $\mathcal{C}$. In the simplest case, codepoints are organized in a one-dimensional topology, typically along a line. In practice, the topology is usually two-dimensional, i.e., initial codepoints are placed in a rectangular or hexagonal grid. Three or more-dimensional topologies are rarely used.

The objective of the SOM algorithm is to fit the map to a data set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ while preserving the predefined topology of the codepoints. On the one hand, the concept of "fitting" suggests that the algorithm minimize a global distortion function or some sort of average distance between the data points and their projections. On the other hand, preserving the topology means that some of the accuracy of the quantization is traded for keeping the smoothness of the topological mapping. In an ideal situation the two criteria can be combined into an objective function which is then minimized by an algorithm. Although in some special cases such objective functions can be defined, in general, no such function exists for the SOM algorithm.

Similarly to all algorithms presented in this chapter, the SOM algorithm alternates between a projection and an expectation step. The projection step is identical to the projection step of the GL algorithm, i.e., each data point is placed into the Voronoi-set of its nearest codepoint. In the expectation step the codepoints are relocated. In the GL algorithm, the new codepoint $\mathbf{v}_\ell$ is set to the center of gravity of the corresponding Voronoi-set V_ℓ. In the SOM algorithm, the new codepoint is a weighted average of *all* data points where the weight of a data point $\mathbf{x}_i$ in effecting the update of $\mathbf{v}_\ell$ depends on how close it projects to $\mathbf{v}_\ell$. Here, "closeness" is measured in the topology defined by $\mathbf{W}$. Formally, let $\ell(i)$ denote the index of the nearest codepoint to $\mathbf{x}_i$ in the jth iteration, that is,

$$\ell(i) = \operatorname*{arg\,min}_{\ell=1,\ldots,k} \left\| \mathbf{v}_\ell^{(j)} - \mathbf{x}_i \right\|^2.$$

Then the new codepoint is given by

$$\mathbf{v}_\ell^{(j+1)} = \frac{\sum_{i=1}^n w_{\ell,\ell(i)} \mathbf{x}_i}{\sum_{i=1}^n w_{\ell,\ell(i)}}.$$

Although there exists no theoretical proof of the convergence of the algorithm, in practice it is observed to converge. Since there is no known objective function that is minimized by the iteration of the two steps, convergence of the algorithm is, in theory, difficult to determine. The average distortion (7) used in vector quantizer design is guaranteed to decrease in the projection step but

may increase in the expectation step. The weighted average distortion defined by

$$\Delta_n(\mathcal{C},\mathbf{W}) = \sum_{\ell=1}^{k} \frac{\sum_{i=1}^{n} w_{\ell,\ell(i)} \left\|\mathbf{v}_\ell - \mathbf{x}_i\right\|^2}{\sum_{i=1}^{n} w_{\ell,\ell(i)}}$$

is minimized in the expectation step but may increase in the projection step. In the formal description of the algorithm below we use the "general" distance function Δ indicating that the exact convergence criteria is unknown.

Algorithm 5 (The SOM algorithm)

Step 0 *Set* $j = 0$*, and set* $\mathcal{C}^{(0)} = \left\{\mathbf{v}_1^{(0)},\ldots,\mathbf{v}_k^{(0)}\right\}$ *to an initial codebook.*

Step 1 (Partition) *Construct* $\mathcal{V}^{(j)} = \left\{V_1^{(j)},\ldots,V_k^{(j)}\right\}$ *by setting*
$V_i^{(j)} = \left\{\mathbf{x} : \Delta\left(\mathbf{x},\mathbf{v}_i^{(j)}\right) \le \Delta\left(\mathbf{x},\mathbf{v}_m^{(j)}\right),\ m = 1,\ldots,k\right\}$ *for* $i = 1,\ldots,k$.

Step 2 (Expectation) *Construct* $\mathcal{C}^{(j+1)} = \left\{\mathbf{v}_1^{(j+1)},\ldots,\mathbf{v}_k^{(j+1)}\right\}$ *by setting*
$\mathbf{v}_\ell^{(j+1)} = \frac{\sum_{i=1}^{n} w_{\ell,\ell(i)}\mathbf{x}_i}{\sum_{i=1}^{n} w_{\ell,\ell(i)}}$ *for* $\ell = 1,\ldots,k$ *where* $\ell(i) = \arg\min_{\ell=1,\ldots,k} \left\|\mathbf{v}_\ell^{(j)} - \mathbf{x}_i\right\|^2$.

Step 3 *Stop if* $\left|1 - \frac{\Delta^{(j+1)}}{\Delta^{(j)}}\right|$ *is less than a certain threshold. Otherwise, let* $j = j+1$ *and go to Step 1.*

Note that if the weight matrix $\mathbf{W}$ is the identity matrix, the SOM algorithm is identical to the GL algorithm. In practice, the neighborhood width of the codepoints is usually decreased as the optimization proceeds. In the final steps of the algorithm, $\mathbf{W}$ is usually set to the identity matrix, so the SOM and the GL algorithms are equivalent at this point, however, this does not imply that the resulting final codebooks generated by the algorithms are equivalent.

To illuminate the connection between self-organizing maps and principal curves, consider the HS algorithm with a locally weighted running-line smoother used for local averaging. In the expectation step of the HS algorithm, an $n \times n$ weight matrix is defined by (39) where the weight $w_{\ell,m}$ determines the effect of $\mathbf{x}_m$ in estimating the curve at the projection point of $\mathbf{x}_\ell$. Now consider the SOM algorithm with $k = n$ codepoints running side by side with the HS algorithm on the same data set $\mathcal{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$. Assume that after the jth iteration the n projection points to the principal curve $\mathbf{f}^{(j)}\left(t_1^{(j-1)}\right),\ldots,\mathbf{f}^{(j)}\left(t_n^{(j-1)}\right)$ are identical to the n codepoints $\left\{\mathbf{v}_1^{(j)},\ldots,\mathbf{v}_n^{(j)}\right\}$ of the SOM, and that the weight matrix $\mathbf{W}$ of the SOM is defined by (39). In this case the estimation procedures in the following expectation steps of the two algorithms are almost identical. The only difference is that the HS algorithm uses weighted least square regression, while the SOM algorithm applies weighted average in computing the new codepoint.

This practically negligible difference originates from a more important conceptual difference, namely, that the objective of the HS algorithm is to find an optimal *curve*, whereas the SOM algorithm optimizes a *set of vertices* equipped with a one-dimensional topology (in this case). The

practical similarity of the actual methods then emerges from following two facts. First, the HS algorithm approximates the curve by a polygonal curve so the task is then to optimize the vertices of the polygonal curve. Second, the codepoints produced by the SOM algorithm, when depicted, are usually connected by line segments. The connections here are based on the neighborhood relations generated by the weight matrix **W** (such that each "inner" codepoint is connected to its two nearest neighbors, while each "endpoint" is connected to its nearest neighbor), and serve as a tool to visualize the topology. The line segments are not by any means part of the manifold fitted to the data. This conceptual difference is also the source of a major practical difference of the two methods when we consider the *entire optimization*, not only one projection step for which a scenario described above created artificially. This major difference is that the weights of the SOM algorithm are either kept unchanged during the optimization or they are modified deterministically in a data-independent fashion (i.e., the neighborhoods of the codepoints are shrunk as described above in connection with the GL algorithm), whereas the weights (39) of the HS algorithm are reset in every iteration based on the relative positions of the projections points.

We note here that the conceptual difference between principal curves and self-organizing maps will result in a major practical difference between the SOM algorithm and the polygonal line algorithm to be introduced in Chapter 5 for estimating principal curves of data sets. This practical difference and its implications will be discussed in Section 5.1.9.

Limitations of the SOM Algorithm and Principled Alternatives

Despite its simplicity and efficiency, the SOM algorithm has several weaknesses that make its theoretical analysis difficult and limit its practical usefulness. The first and probably most important limitation of the SOM algorithm is that there does not exist any objective function that is minimized by the training process as showed by Erwin et al. [EOS92]. Not only has this limitation theoretical consequences, namely that it is hard to show any analytical properties of the resulting map, but it also makes experimental evaluation difficult. Nevertheless, several recent studies attempt to objectively compare the SOM algorithm to other methods from different aspects. These studies suggest that it is hard to find any criteria under which the SOM algorithm performs better than the traditional techniques used for comparison.

In a study on the efficiency of the SOM algorithm for *data clustering*, Waller et al. [WKIM98] compared the SOM algorithm to five different clustering algorithms on 2850 artificial data sets. In these experiments, zero neighborhood width was used in the final iterations of the SOM algorithm, consequently, it was found that the SOM and the *k*-means clustering algorithms (the stochastic version of the GL algorithm) performed equally well in terms of the number of misclassified data points (both being better than the other hierarchical clustering methods). The significance of this

result is that the nonzero neighborhood width applied in the beginning of the SOM iteration does not improve the clustering performance of the SOM algorithm. It was also shown by Balakrishnan et al. [BCJL94], who compared the SOM algorithm to *k*-means clustering on 108 multivariate normal clustering problems, that if the neighborhood width does not decrease to zero, the SOM algorithm performs significantly worse than the *k*-means clustering algorithm.

Evaluating the *topology preservation* capability of the SOM algorithm, Bezdek and Nikhil [BP95] compared the SOM algorithm to traditional multidimensional scaling techniques on seven artificial data sets with different numbers of points and dimensionality, and different shapes of source distributions. The degree of topology preservation of the data was measured via a Spearman rank correlation between the distances of points in the input space and the distances of their projections in the two-dimensional space. [BP95] found that the traditional statistical methods preserve the distances much more effectively than the SOM algorithm. This result was also confirmed by Flexer [Fle99].

In an empirical study on SOM's ability to do *both clustering and topology preservation* in the same time, Flexer [Fle97, Fle99] compared the SOM algorithm to a combined technique of *k*-means clustering plus Sammon mapping [Sam69] (a traditional statistical method used for multidimensional scaling) on the cluster centers. If zero neighborhood width was used in the final iterations of the SOM algorithm, the SOM algorithm performed almost equally well to the combined algorithm in terms of the number of misclassified data points (confirming the results of [WKIM98]). However, the SOM algorithm performed substantially worse than the combined method in preserving the topology as a consequence of the restriction of the planar grid topology of the SOM. Using a nonzero neighborhood width at the end of the training did not improve the performance of the SOM algorithm significantly.

There have been several attempts to overcome the limitations of the SOM algorithm. Here we briefly describe two alternative models which we selected on the basis of their strong connection to principal curves. The Generative Topographic Mapping (GTM) of Bishop et al. [BSW98] is a principled alternative to SOM. Similarly to Tibshirani's semi-parametric model [Tib92] described in Section 3.2.1, it is assumed that the data was generated by adding an independent Gaussian noise to a vector generated on a nonlinear manifold according to an underlining distribution. To develop a model similar in spirit to the SOM and to make the optimization problem tractable, the latent manifold is assumed to be a set of points of a regular grid. In this sense the GTM can be considered as a "discretized" version of Tibshirani's model (in which the nonlinear manifold is a curve). Similarly to Tibshirani's algorithm, the yielded optimization problem is solved by an EM algorithm. An interesting relationship between the HS algorithm, Tibshirani's algorithm, the GTM algorithm and our polygonal line algorithm is pointed out in Section 5.1.9, after the latter method is described.

To overcome the limitations of the SOM algorithm caused by the predefined topology of the cluster centers, Balzuweit et al. [BDHW97, DBH96] proposed a method to adaptively modify the topology during the training process. The basic idea is to set the weight w_{ij} proportional to the number of data points whose nearest and the second nearest neighbors are the cluster centers $\mathbf{v}_i$ and $\mathbf{v}_j$. The resulting dynamic topology is similar to the topology induced by the local averaging rule in the HS algorithm. The main advantage of the method is that it allows the formation of loops and forks during the training process as opposed to the single curve topology of the HS algorithm.

3.2.3 Nonlinear Principal Component Analysis

In the nonlinear PCA model of Kramer [Kra91] the empirical loss minimization principle described in Section 1.1.3 is slightly modified. According to the principle formalized in (3), given a data set $\mathcal{X}_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ and a set $\mathcal{F}$ of curves[2], we pick the curve that minimizes the average distance between the data points and the curve. Formally, we minimize

$$\sum_{i=1}^{n} \Delta(\mathbf{x}_i, \mathbf{f}) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{f}(t(\mathbf{x}_i))\|^2 \tag{45}$$

over all curves $\mathbf{f} \in \mathcal{F}$ where the projection index $t(\mathbf{x}_i) = t_{\mathbf{f}}(\mathbf{x}_i)$ is a *fixed* function of $\mathbf{f}$ and $\mathbf{x}_i$ defined in (14). On the other hand, in nonlinear PCA the projection index is also subject to optimization, i.e., we minimize (45) with respect to all functions $\mathbf{f} \in \mathcal{F}$ and $t \in \mathcal{T}$. The function classes $\mathcal{F}$ and $\mathcal{T}$ contain continuous smooth functions tailored to the gradient-based optimization method usually used to carry out the optimization of (45). In particular, [Kra91] uses functions of the form

$$t(\mathbf{x}_i) = \sum_{j=1}^{k_1} w_j^{(1)} \sigma\left(\mathbf{w}_j^{(2)} \mathbf{x}_i + b_j^{(2)}\right)$$

and

$$f_i(t) = \sum_{j=1}^{k_2} w_j^{(3)} \sigma\left(w_j^{(4)} t + b_j^{(4)}\right), i = 1, \ldots, n$$

where σ is any continuous and monotonically increasing function with $\sigma(x) \to 1$ as $x \to +\infty$ and $\sigma(x) \to 0$ as $x \to -\infty$, and (45) is optimized with respect to the unknown parameters $w_j^{(1)}, b_j^{(2)} \in \mathbb{R}, \mathbf{w}_j^{(2)} \in \mathbb{R}^d, j = 1, \ldots, k_1$ and $w_j^{(3)}, w_j^{(4)}, b_j^{(4)} \in R, j = 1, \ldots, k_2$. Comparing nonlinear PCA to principal curves, Malthouse et al. [MMT95] pointed out that the main difference between the two models is that principal curves allow the projection index $t(\mathbf{x})$ to be discontinuous at certain points.

[2]The original model of [Kra91] is more general in the sense that it allows arbitrary-dimensional manifolds. Our purpose here is to compare nonlinear PCA to principal curves, so, for the sake of simplicity and without loss of generality, we describe nonlinear PCA as a curve-fitting method.

The required continuity of the projection index causes the nonlinear PCA optimization to find a sub-optimal solution $(\hat{\mathbf{f}}, \hat{t})$ in the sense that in general, the projection of a point $\mathbf{x}$ will not be the nearest point of $\hat{\mathbf{f}}$ to $\mathbf{x}$, i.e.,

$$\|\mathbf{x} - \hat{\mathbf{f}}(\hat{t}(\mathbf{x}))\| > \inf_t \|\mathbf{x} - \hat{\mathbf{f}}(t)\|.$$