Local Principal Curves

Jochen Einbeck Gerhard Tutz Ludger Evers
{einbeck,tutz,ludger}@stat.uni-muenchen.de
Ludwig-Maximilians-Universität München, Akademiestraße 1,
80799 München, Germany

8th May 2003

Abstract

Principal components are a well established tool in dimension reduction. The extension to principal curves allows for general smooth curves which pass through the middle of a $p-$ dimensional data cloud. In this paper local principal curves are introduced, which are based on the localization of principal component analysis. The proposed algorithm is able to identify closed curves as well as multiple curves which may or may not be connected. For the evaluation of performance of data reduction obtained by principal curves a measure of coverage is suggested. The selection of tuning parameters is considered explicitely yielding an algorithm which is easy to apply. By use of simulated and real data sets the approach is compared to various alternative concepts of principal curves.

Key Words: Local smoothing, mean shift, principal components, principal curves.

1 Introduction

The classical problem of how to find the best curve passing through some data points $(x_i, y_i), i = 1, \ldots, n$ can be handled in two fundamentally different ways. Let us regard the data points as realizations of i.i.d. random variables (X_i, Y_i) drawn from a population (X, Y). A common approach is to regard X as an explanatory variable for the dependent variable Y. This concept is used in all methods where the focus is on regression or smoothing and is especially useful when the objective is prediction of the dependent variable from the observations x_i. Thereby X and Y have an asymmetric relationship and cannot be interchanged without changes of the results.

In contrast, X and Y may be regarded as symmetric, thus we do not assume that one variable can be made reponsible for the value of the other one. Rather they are generated simultanuously from a common underlying distribution. These approaches are useful when the focus is on dimension reduction or simply description of the data. Representants here are methods like the ACE algorithm, canonical correlation or principal component analysis. Linear Principal components, introduced by Pearson (1901), are a common tool in multivariate analysis, applied for example in feature extraction or dimension reduction. Jolliffe (1986) gave an overview on properties and applications of principal components. Nonlinear principal components have been developed by Schölkopf & Smola (1998) and successfully employed for pattern recognition.

A natural extension of principal components are principal curves, which are descriptively defined as one-dimensional smooth curves that pass through the "middle" of a $p-$dimensional data set. Though this concept is intuitively clear, there is much flexibility in how to define the "middle" of a distribution or a data cloud. Hastie & Stuetzle (1982) (hereafter HS), who did the groundbreaking work on principal curves, use the concept of self-consistency (Tharpey & Flury, 1996), meaning that each point of the principal curve is the average of all points that project there. A variety of other definitions of principal curves have been given subsequently by Tibshirani (1992), Kégl, Krzyzak, Linder & Zeger (2000) (hereafter KKLZ), and more recently Delicado (2001), which differ essentially in how the "middle" of the distribution is found.

Apart from Delicado (2001), all concepts mentioned above work more or less as follows: They start with a straight line, which is mostly the first principal component of the data set, and try to dwell out this line or concatenate other lines to the initial line until the resulting curve is passing satisfactory through the middle of the data. This methodology leads to some technical problems. HS generally exclude intersecting curves from the definition of principal curves and are not able to handle closed curves. Banfield & Raftery (1992) (hereafter BR) provide a bias corrected version of the HS algorithm which solves the latter problem, but yields more wiggly results than HS. Chang & Ghosh (1998) combine the algorithms of HS and BR and show that this yields a smooth and unbiased principal curve, at least for simple data situations. Tibshirani's theoretically attractive approach seems to have the same problems as HS, though not explicitely stated, and further seems to have a lack of flexibility for strongly skewed data. These difficulties have been solved by Verbeek, Vlassis & Kröse (2001), but at the expense of an apparent wiggly principal curve, since polygonal lines are connected in a somehow unsmooth manner. KKLZ work also with polygonal lines and obtain with high computational effort a smooth and flexible principal curve, which only fails for very complicated data structures. None of these algorithms seems to be able to handle curves which consist of some multiple or disconnected parts. Recently, Kégl & Krzyzak (2002) provided a promising algorithm to obtain *principal graphs*, i.e. multiple connected piecewise linear curves, in the context of skeletonization of hand-written characters.

All these methods have to be regarded as global, since in every step of their algorithms, or at least in the initial step, all available data points are used. As alternative to the global methods, which lead to exploding computational costs for large data sets or high-dimensional data, it would be desirable to have a local method at hand, which only considers data which are close to the target point. Lately, Delicado (2001) proposed the first principal curve approach which can be called local. Assume a d-dimensional random vector X and n random samples $X_i \in \mathbb{R}^d, i = 1, \ldots, n$ from X, where $X_i = (X_{i1}, \ldots, X_{id})$. For each point x, Delicado considers the hyperplane $H(x, b)$ which contains x and is orthogonal to a vector b. The set of vectors $b^*(x)$ minimizing the total variance $\phi(x, b) = TV(X|X \in H(x, b))$ defines a function $\mu^*(x) = E(X|X \in H(x, b^*(x)))$. Principal oriented points (POPs) are introduced as fix points of the function $\mu^*(\cdot)$. For a suitable interval $I \in \mathbb{R}$, α is called a principal curve of oriented points (PCOP) if $\{\alpha(s)|s \in I\}$ is a subset of the fix point set of μ^*. Delicado shows that POPs exist, and that in case $b^*(x)$ is unique (this implies that the principal curve is a function), to each POP exists a PCOP passing through it. Since the hyperplanes H are sets of measure zero, it is necessary to employ a kind of smoothing for calculating the conditional expectation on the hyperplane. This is achieved by projecting all data points on $H(x, b)$, obtaining points X_i^H, and assigning weights

$$w_i = w(|(X_i - x)^T b|), \tag{1}$$

where w is a decreasing positive function, e.g. $w(d) = K(d/h)$, with a kernel function K. Let $\tilde{\mu}(x, b)$ denote the weighted expectation of the X_i^H with weights w_i. Now $\mu^*(x)$ is approximated by $\tilde{\mu}^*(x) = \tilde{\mu}(x, \tilde{b}^*(x))$, where $\tilde{b}^*(x)$ (and hence H) is constructed such that the variance of the projected sample, weighted with w_i, is minimized. Localization enters here twofold. Firstly, by applying (1), points near to the hyperplane are upweighted. Secondly, a cluster analysis is performed on the hyperplane, and only data in the local cluster are considered for averaging. How is the principal curve found in practice? The algorithm searches the fix point set of $\tilde{m}^*(x)$ as follows. Repeatedly, choose a point randomly from the sample $X_1, \ldots, X_n$ and call it $x_{(0)}$. Then iterate $x_{(\ell)} = \tilde{\mu}^*(x_{(\ell-1)})$ until convergence. In this manner a finite set of POPs is obtained. However, no fix point theorem guarantees convergence of this algorithm, although Delicado reports quick convergence for some real data sets. In order to obtain a PCOP from a set of POPs, Delicado proposes an idea which we will further exploit. Assume an POP x_1 calculated as explained. From the set of principal directions $\tilde{b}^*(x_1)$, choose one vector b_1. Now walk a step of length ∂ from x_1 in direction of b_1, i.e.

$$x_2^0 = x_1 + \partial b_1, \tag{2}$$

where ∂ is previously fixed. The point x_2^0 serves as a new starting point for a new iterating process, leading to a new point x_2 of the principal curve. This is repeated k times until no points X_i can be considered to be near the hyperplane $H(x_k^0, b_k)$. Then return to (x_1, b_1) and complete the principal curve in direction of $-b_1$. Afterwards move on to another of the previously chosen POPs and continue analogously.

Delicado's concept is mathematically elegant, theoretically well elaborated, and works fine in the examples he provided. It might work fine even for complicated data structures (spirals, disconnected branches, etc.), though he didn't provide examples for those cases. One might consider it as a drawback that the concept is mathematically demanding and not intuitively clear. Further, Delicado does not reflect the choice of parameters ∂ and h.

In this paper, we introduce a concept similar to that of Delicado. However, we replace the fix points of $\tilde{\mu}^*$ by local centers of mass, and replace the principal direction b_1 by a local principal component. We call the resulting curve, which consists of a series of local centers of mass, *local principal curve*. We introduce the notion of *coverage*, which evaluates the performance of the principal curve approximation and is a helpful tool to compare the performance of different principal curve algorithms. We show that, using this concept of coverage, the parameters which are necessary for our algorithm can easily be selected in a data-adaptive way. The price paid for the easiness of the concept is that in contrast to Delicado's approach there is no statistical model and consequently it is hard to derive theoretical results. However, in Section 5 we give a theoretical justification for our method. The algorithm will be presented in the following section.

2 Local Principal Components

Assume a d-dimensional data cloud $X_i \in \mathbb{R}^d, i = 1, \ldots, n$, where $X_i = (X_{i1}, \ldots, X_{id})$. We try to find a curve which passes through the "middle" of the data cloud. The curve will be calculated by means of a series of local centers of mass of the data, according to the following strategy:

1. Choose a suitable starting point $x_{(0)}$. Set $x = x_{(0)}$.
2. Calculate the local center of mass μ^x around x.
3. Perform a principal component analysis locally at x.
4. Find the new value x by following the first local principal component γ^x starting at μ^x.
5. Repeat steps 2 to 4 until μ^x remains (approximately) constant.

The series of the μ^x make up the desired curve. In the sequel we will explain these steps in detail:

1. Selection of the starting point

In principle, every point $x_{(0)} \in \mathbb{R}^d$ which is in or close to the data cloud can be chosen as starting point. There are two ideas which suggest themselves:

- Based on a density estimate the point with the highest density $x_{(0)} = max_{x \in \mathbb{R}} \hat{f}(x)$ is chosen.
- A point $x_{(0)} = X_i$ is chosen at random from the set of observations.

The advantage of the density method is that one can be quite sure not to start in a blind alley, whereas a randomly chosen point could be an outlier far from the data cloud which stops the algorithm already in the first loop. However, this is not very likely, and the computational costs of the second approach are much lower. Moreover, for handling crossings a randomly chosen starting point is even superior to a high density point.

2. Calculating the local center of mass

Let H be a bandwidth matrix and $K_H(\cdot)$ a $d-$ dimensional kernel function. Given that all components of X are measured on the same scale, we set $H = \{h^2 \cdot I : h > 0\}$, with I standing for the d-dimensional identity matrix. For a detailed description of multivariate kernels and bandwidth matrices see Wand & Jones (1993). For selection of h, see Section 7. The local center of mass around x is given by

$$\mu(x) = \frac{\sum_{i=1}^{n} K_H(X_i - x) X_i}{\sum_{i=1}^{n} K_H(X_i - x)} \tag{3}$$

This estimator and its relation to the Nadaraya-Watson estimator have been analyzed in Comaniciu & Meer (2002). For ease of notation, we will abbreviate $\mu^x = \mu(x)$ in the following. We denote by μ_j^x the j-th component of $\mu(x)$.

3. Calculating the local principal component

Let $\Sigma^x = (\sigma_{jk}^x)$ denote the local covariance matrix of x, whoose (j,k)-th entry $(1 \leq j, k \leq d)$ is given by

$$\sigma_{jk}^x = \sum_{i=1}^{n} k_i (X_{ij} - \mu_j^x)(X_{ik} - \mu_k^x) \tag{4}$$

with weights $k_i = K_H(X_i - x) / \sum_{i=1}^{n} K_H(X_i - x)$, and H as in 2. Let γ^x be the first eigenvector of Σ^x. Then γ^x is the first column of the loadings matrix Γ^x from the principal components decomposition $(\Gamma^x)^T \Sigma^x \Gamma^x = \Lambda^x$, where $\Lambda^x = (\lambda_1^x, \ldots, \lambda_p^x)$ is a diagonal matrix containing the ordered eigenvalues of Σ^x, with $\lambda_1^x \geq \ldots \geq \lambda_p^x$.

Note that the denotation "local principal components" is not new, but has been previously used for linear principal components localized in clusters (Skarbek, 1996; Kambhatla & Leen, 1997) or based on contiguity relations (Aluja-Banet & Nennell-Torrent, 1991) rather than by kernel functions.

4. Obtaining an updated value

The local principal component line v^x can now be parameterized as

$$v^x(t) = \mu^x + t\gamma^x \quad (t \in \mathbb{R}), \tag{5}$$

and we obtain an updated value of x by setting

$$x := \mu^x + t_0 \gamma^x, \tag{6}$$

in analogy to step (2) of Delicado's algorithm. A suitable value of t_0 thereby has to be chosen beforehand. We defer the task of how to select t_0 to Section 7.

5. Stop when μ^x remains constant

When the end of the data cloud is reached, the algorithm will naturally get stuck and produce approximately constant values of μ^x. One might stop before this state occures, e.g. when the difference between previous and current center of mass falls below a certain threshold.

The mechanism is demonstrated in Fig. 1. The starting point $x_{(0)}$ is denoted by 0. The radius of the circle is equal to the bandwith $h = 0.2$. Calculating the local center of mass around 0 yields the nearby point m. Moving along the first principal component with $t_0 = 0.2$ leads to the new point x denoted by "1", and so on. The series of m's is the local principal curve. Note that the algorithm is based on finding an equilibration between opposing tendencies: On the one hand, the local principal components are oversteering, i.e. tending "outside" to the concave side of the curvature of the data cloud. On the other hand, the calculation of the local center of mass is smoothing the data towards the interior and thus in the opposite direction. These two effects together ensure that the estimated principal curve is not systematically biased.

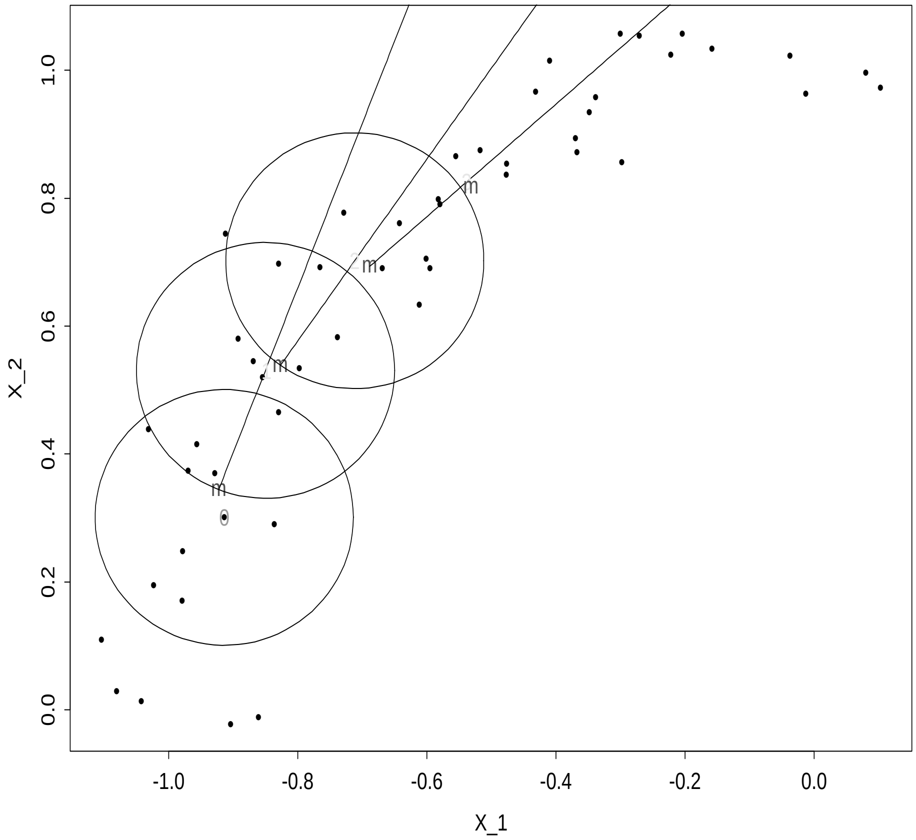


Figure 1: Demonstration of the local principal curve algorithm.

3 Technical details

In practice, some modifications of the above algorithm are useful, which we describe in the following.

3.1 Maintainig the direction

A principal component line always has two directions, thus the corresponding eigenvector γ^x could be replaced by its negative value $-\gamma^x$. Depending on the orientation of the eigenvector, the constructed curve moves in opposite directions. If this direction changes from one step to another, the algorithm dangles between these two points and will never escape. Therefore one should check in every step that the local eigenvector has the same direction as in the previous step. This can be done by calculating the angle $\alpha^x_{(i)}$ between the eigenvectors $\gamma^x_{(i-1)}$ and $\gamma^x_{(i)}$ belonging to the $(i-1)$-th resp. i-th step , which is given by

$$\cos(\alpha^x_{(i)}) = \gamma^x_{(i-1)} \circ \gamma^x_{(i)},$$

where $\circ$ denotes the scalar product. If $\cos(\alpha^x_{(i)}) < 0$, set $\gamma^x_{(i)} := -\gamma^x_{(i)}$, and continue the algorithm as usual. This "signum flipping" has been applied in the step from "2" to "3" in Figure 1.

3.2 Running backwards from $x_{(0)}$

When one starts at a point $x_{(0)}$ and moves by means of local principal components to one "end" of the cloud, one has omitted to consider the part between the starting point and the other end of the cloud, except if the data describe a closed curve, e.g. a circle or a ellipse. Therefore it is advisable to run from the starting point in both directions of the first principal component, what in practice means adding a 6th step to the algorithm:

6. For the starting direction $-\gamma^x_{(0)} := -\gamma^{x_{(0)}}$, perform steps 4 and 5.

3.3 Angle penalization

If the data cloud locally forms crossings, at each crossing the local principal curve has three possibilities where to move on. Often one desires that the curve goes straight on at each crossing, and does not turn arbitrarily to the left or right. In order to achieve this effect, we recommend to perform an angle penalization in addition to the signum flipping in each step of the algorithm. This might be done as follows:

Let k be a positive number. For the angle $\alpha^x_{(i)}$, set

$$a^x_{(i)} := |\cos(\alpha^x_{(i)})|^k$$

and correct the eigenvectors according to

$$\gamma^x_{(i)} := a^x_{(i)} \cdot \gamma^x_{(i)} + (1 - a^x_{(i)}) \cdot \gamma^x_{(i-1)}$$

Thus, the higher the value of k, the more the curve is forced to move straight on. We recommend to set set $k = 1$ or 2. For higher values of k the local principal curve looses too much flexibility.

3.4 Multiple initializations

Assume that the data cloud consists of several branches, which might or might not be connected. Then one single local principal curve will fail to describe the whole data set, but will only find one branch. This is a problem inherent to all global principal curve algorithms. In our approach this problem can be solved by doing multiple initializations, i.e. we choose subsequently a series of starting points, so that finally at least one starting point is situated on each branch, and perform the algorithm for each starting point. In this manner the whole data cloud will be covered by the local principal curve. The starting points can be imposed by hand on each of the branches, or, if this is not possible or too cumbersome, they might be chosen randomly. If one has for example two disconnected branches of the data cloud, which contain more or less the same amount of data, then the application of four randomly chosen starting points already effects that with 93.75% probability at least one starting point is on each cloud. For an arbitrary number of branches, Borel-Cantelli's Lemma tells us that with the number of starting points increasing to infinity, each branch is visited with probability 1. In practice this technique proves to work satisfactory, even for a high number of branches. To conclude, for a set of starting points S_0, we add a 7th step to the algorithm:

7. If $S_0 \neq \emptyset$, choose (without replacement) a new starting point $x_{(0)} \in S_0$ and start again with step 1.

It should be noted that our algorithm is deterministic given the starting points, but yields different principal curves for different starting points. However, since in each case the local centers of mass of the same data are calculated, differences of principal curves on the same branch are usually neglectable. In contrary, KKLZ's implementation of their algorithm is strongly indeterministic, and that even for equal starting conditions.

4 Examples

4.1 2-dimensional data

Firstly, we compare the results of our algorithm with some standard examples which were also examined by KKLZ (In this and the following examples, the curves from KKLZ and BR are obtained via the Principal Curves Java program from Balázs Kégl, available at `http://www.iro.umontreal.ca/~kegl/research/pcurves/`. The HS curves were obtained by Hastie's Splus function `http://lib.stat.cmu.edu/S/principal.curve`). We start with a circle with radius $r = 1$, which is contaminated with bivariate uncorrelated Gaussian noise with variance 0.04 in each component. The result is demonstrated in Fig. 2.

We notice that only the BR and the proposed local principal curve (hereafter: LPC) algorithm produce a closed curve, whereas HS and KKLZ lead to an open curve. The LPC curve seems to be a bit wiggly in comparison to the other curves, but it should be noted that the LPC approach is fully nonparametric and is only steered by the data, but not by an initial line like the other approaches. This leads to more flexibility (looking at the data, the bump in the left top is not unlikely to be a real feature of the distribution) at the price of a higher variance.

Secondly, we examine the spiral data from KKLZ, Fig. 10, b) and c) (where the contaminated big spiral is newly simulated). The standard deviation of the noise is equal to 0.01 for both spirals, and in in each experiment 1000 data points were generated. The small spiral, see Fig. 3, is found nearly perfectly by KKLZ and LPC, however the HS algorithm shows a fairly bad performance here. The big spiral is only found by LPC. KKLZ's polygonal line algorithm fails here and yields erratic results, which differ in each run of the algorithm. The result of HS is even worse (compare KKLZ, page 21, Fig. 11.).

Finally, we consider real data recorded by the Office of Remote Sensing for Earth Resources, Pennsylvania State University, which show the location of floodplains in Beaver

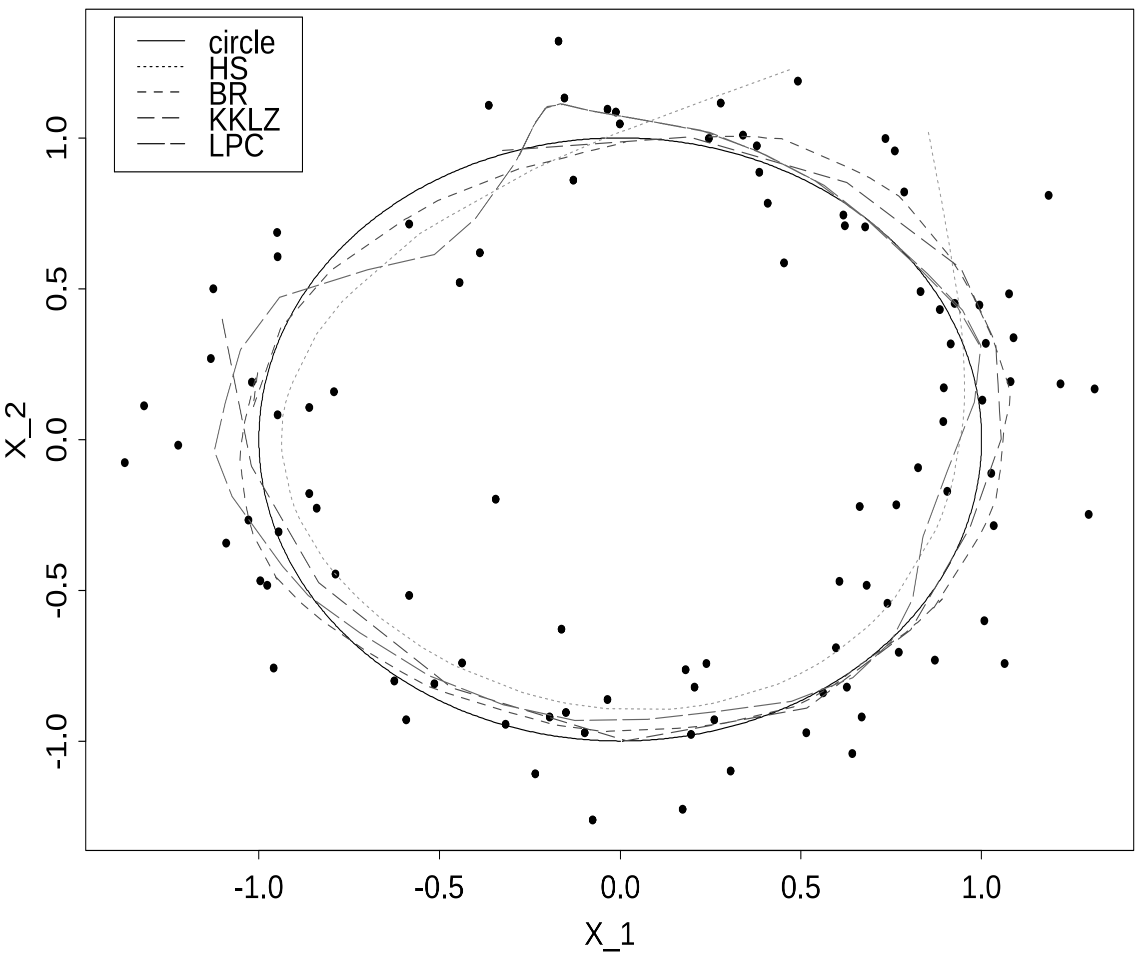


Figure 2: Local principal curve for an underlying circle in comparison to other principal curve algorithms.

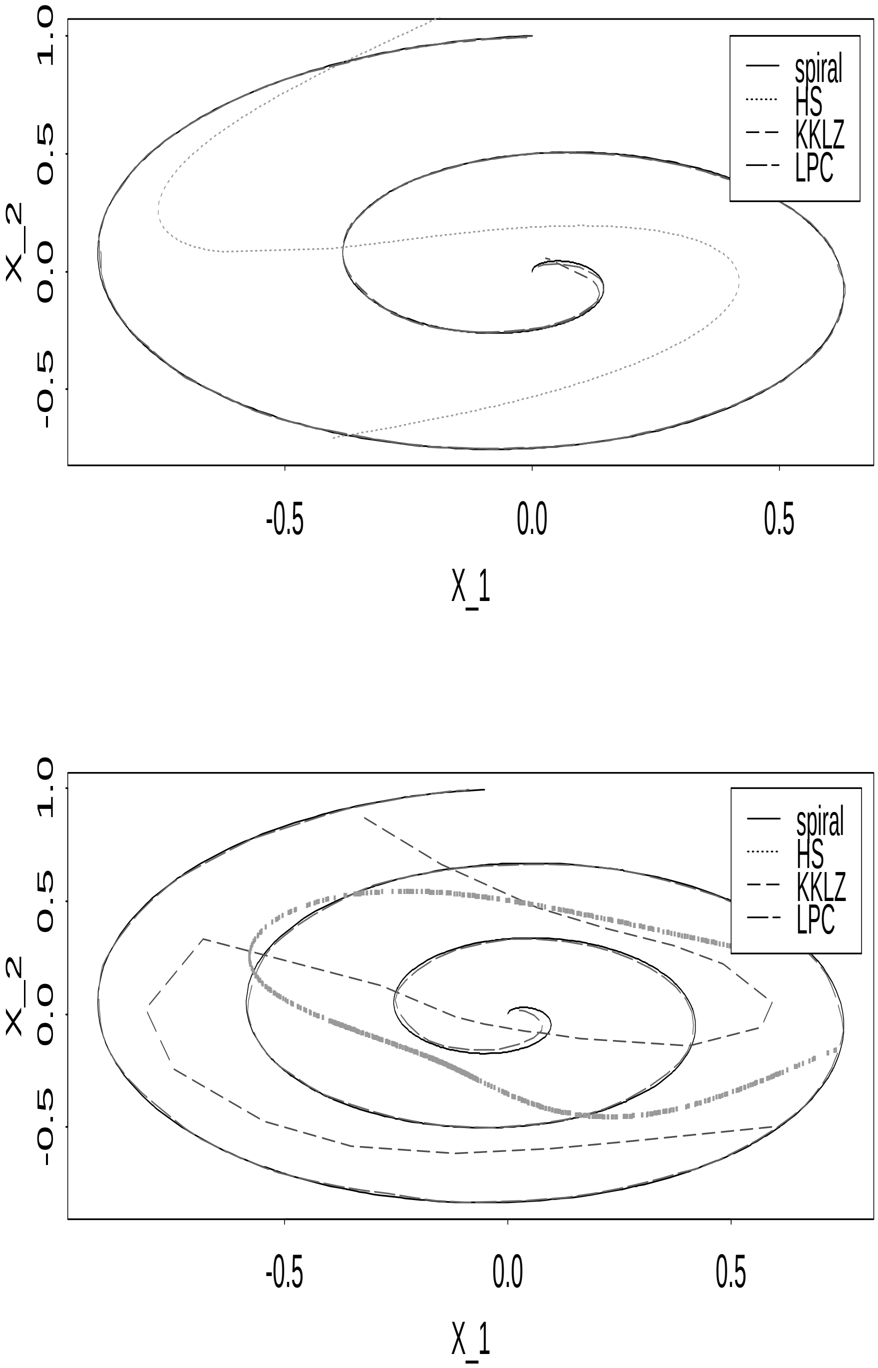


Figure 3: Local principal curve for underlying small and big spirals in comparison to other principal curve algorithms.

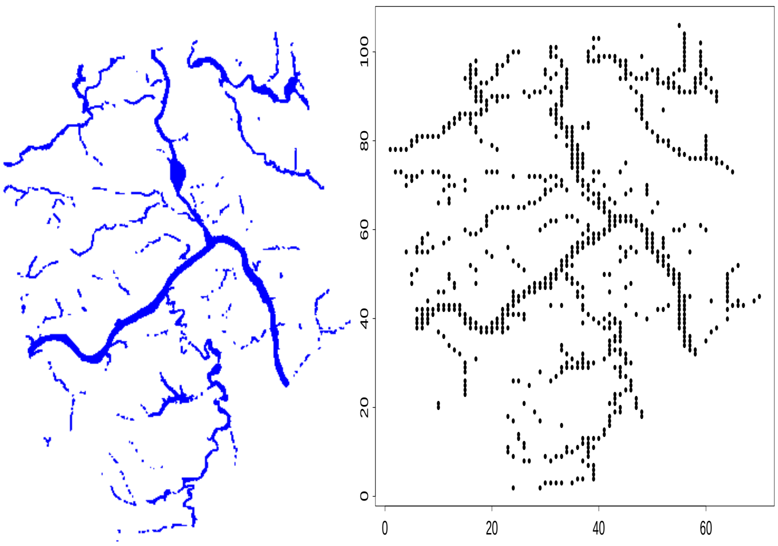


Figure 4: Floodplains in Beaver County, PA. (left: original, right: digitalized).

County, PA, USA, 1996 (Fig. 4). For analyzing the data, we digitalized the map to a grid of $106 \times 70 = 7420$ digits. Fig. 5 shows the result of a run of the LPC algorithm using the digitalized floodplain data. We used 50 initializations and a bandwidth $h = 1.5$ (The automatic selection routine from Section 7 suggests 2.5, but a smaller bandwidth seemed more appropriate in this case.). The principal curve uncovers nicely the principal courses of the floodplains. Taking a look at maps from Beaver county, we see that our principal curve reconstructs the underlying rivers resp. valleys in this district (The data as well as corresponding maps are available at PASDA - Pennsylvania Spatial Data Access, `www.pasda.psu.edu`. The best form to regard those maps is to open the ArcExplorerWeb at `http://www.esri.com/software/arcexplorer` and search in the opening menue for Pennsylvania Spatial Data Access, "PA Streams" or "PA Floodplains"). Note that a quite big cluster in the central bottom is not covered - this simply occurs because none of the randomly chosen starting points is situated there, and this isolated cluster cannot be reached by an external principal curve. More initializations would be necessary to solve this.

4.2 3-dimensional data

We now consider a data set included in the Splus software package, namely the "radial velocity of galaxy NGC7531". This data frame, recorded by Buta (1987), contains the radial velocity of 323 points of that spiral galaxy covering about 200 arc seconds in north-south

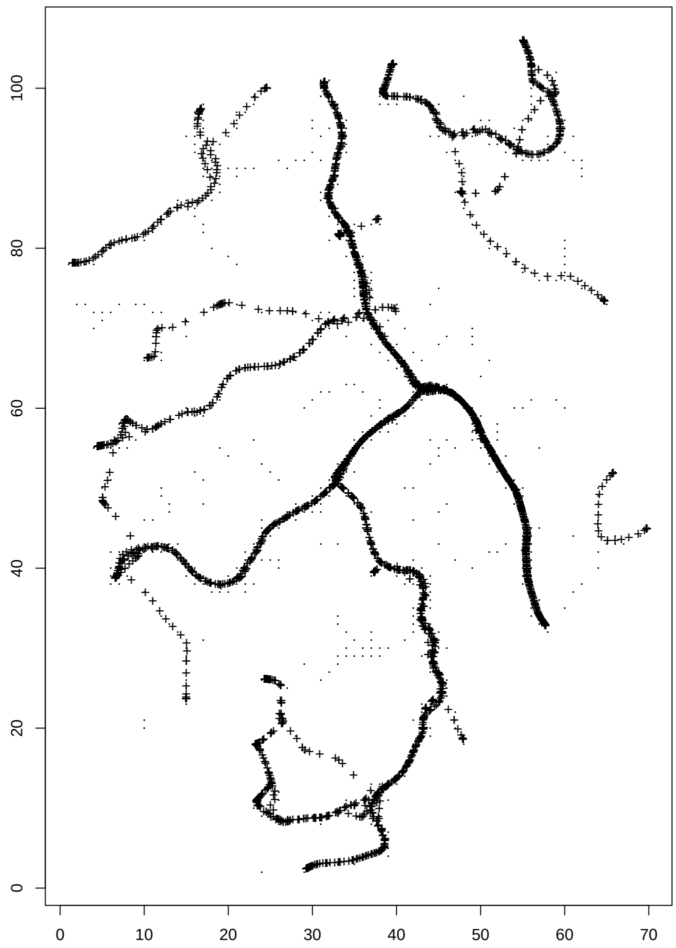


Figure 5: Floodplain data (.) with principal curves (+).

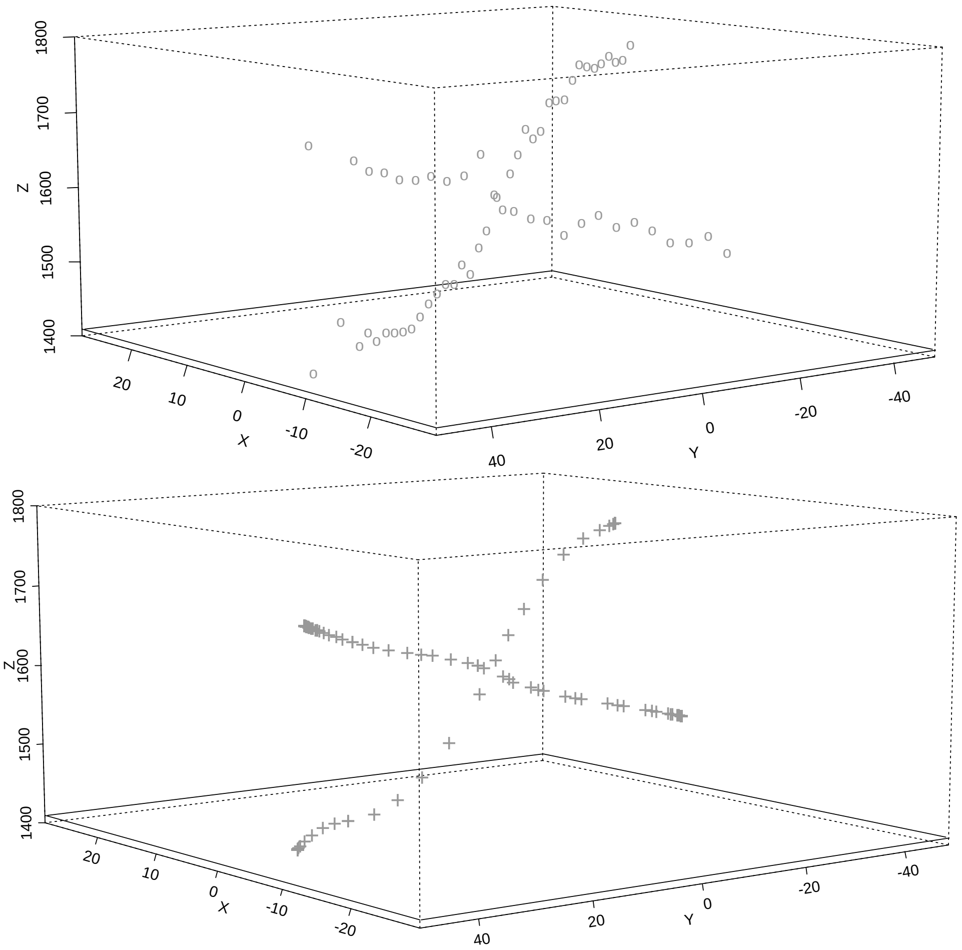


Figure 6: Galaxy data (o) with principal curves (+).

and 135 arc seconds in east-west direction in the celestial sphere. All the measurements lie within seven slots crossing the origin. The x- and y-coordinate describe the east-west resp. north-south coordinate, and the z-coordinate is the radial velocity measured in km/sec. For simplicity, we only consider the first 61 data points of the data set (this corresponds to two slots crossing the origin).

Since the data are now situated on two (connected) branches, we need to inititialize more than once. We choose to initialize 4 starting points. We apply an angle penalization using $k = 2$, which serves to keep the curve on the correct slot at the crossing. The result is shown in Fig. 6.

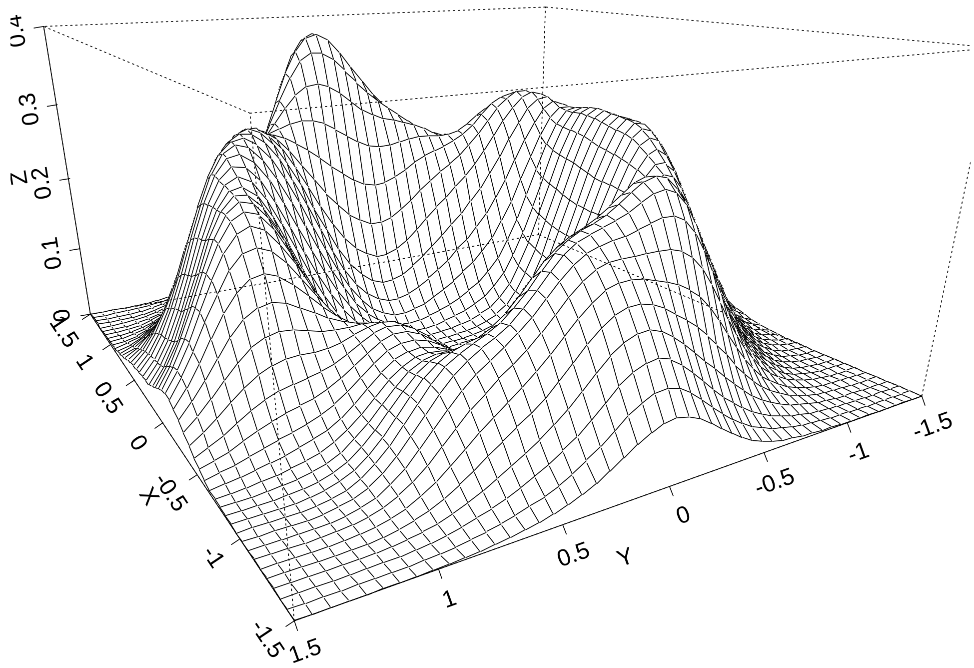


Figure 7: Kernel density estimation of simulated circle data.

5 Theoretical justification

This approach seems to be heuristic to some extent, since we have provided neither a model for the data nor a mathematical precise definition of a local principal curve. In this section we will give some idea which curve we are actually estimating via the LPC algorithm. When we started to work on principal curves, we were not primarily influenced by Delicado's work, but were guided by a simple and appealing idea. It is instructive to take a look at the circle data in Section 4.1. A kernel density estimation yields Figure 7.

Looking at this figure, the course of the principal curve is easily to imagine - one simply has to walk along the crest of the mountain. Unfortunately this crest line, which everybody is able to draw rapidly with a pencil, is mathematically intractable. To our knowledge, there exists no mathematical definition of a crest point. However, we will argue in the following that the principal curve we are estimating by means of our algorithm is approximating this crest line.

Comaniciu, Ramesh & Meer (2001) and Comaniciu & Meer (2002), among others, study the properties of the so-called *mean shift vector*

$$M(x) = \mu(x) - x, \tag{7}$$

with $\mu(x)$ being the local center of mass (3). They provide two results which are of interest for us:

For a Gaussian kernel K and a bandwidth matrix $H = \{h^2 \cdot I : h > 0\}$,

(A) the mean shift vector (7) is proportional to the estimated gradient function $\hat{\nabla} f_K(x)$, where the estimated gradient is the gradient of the kernel density estimator

$$\hat{f}_K(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right), \tag{8}$$

(B) the sequence

$$\begin{aligned} m^{(0)} &= x \\ m^{(k+1)} &= \mu(m^{(k)}) \end{aligned}$$

converges to a nearby point where the estimator (8) has zero gradient, i.e. to a mode candidate of the kernel density.

For other kernels these statements continue to hold under certain conditions, if in (8) K is substituted by its *shadow*, see Comaniciu & Meer (2002).

Let us return to our algorithm now. For any point x, we can calulate a local center of mass $\mu(x)$ via function (3). It is easy to imagine, that the closer x is to the middle of the distribution of the data, the smaller is the mean shift. According to (7) this mean shift is zero for the fix point set of function (3), i.e. the set

$$\{x | \mu(x) = x\}$$

(what shows another analogy of our concept to that of Delicado). Considering (A) we realize that these are the points where the estimated gradient function of the density is minimized, which is the case for modes of the estimated density. By applying (B), we thus have a tool for estimating the modes of the density of the data. This is however not our intention: An algorithm like this would get stuck at the modes and be unable to *connect* the modes in a proper way. Therefore, in each step of the algorithm, we employ only the first loop of the iterative process (B), which brings us near the crests, but not nesessarily in a mode point. Then, for not getting stuck in a mode, we move along a little step in direction of the local principal component (what means in practice: along a crest). If thereby, after one or more steps, a point $x_{(k)}$ is approached which is near to a new mode, then the local center of mass $\mu(x_{(k)})$ will tend to this mode, as the following (quite trivial) lemma shows. If no further modes exists, the algorithm will stop itself when the end of the data cloud is reached.

Lemma 1. *Let $X_i \in \mathbb{R}^d, i = 1, \ldots, n$ be a data cloud and H be a bandwidth matrix. Let μ_0 a fix point of (3) resp. H and $x \longrightarrow \mu_0$. Then, applying the same bandwidth matrix H, we have convergence $\mu(x) \longrightarrow \mu_0$.*

Proof

$$|\mu(x) - \mu_0| = \left| \frac{\sum_{i=1}^n K_H(X_i - x)X_i}{\sum_{i=1}^n K_H(X_i - x)} - \frac{\sum_{i=1}^n K_H(X_i - \mu_0)X_i}{\sum_{i=1}^n K_H(X_i - \mu_0)} \right| \longrightarrow 0$$

for continuous non-zero kernel-functions.

6 Coverage

There is need for some criterion to evaluate the performance of a principal curve. This is usually done by means of a quantitative measure as the expected squared distance

$$\triangle(m) = E\left(\inf_t ||X - m(t)||^2\right) \tag{9}$$

between data X and the curve m. Principal curves according to HS are critical points of (9), whereas principal curves by KKLZ are minimizing (9) over a class of curves with bounded length. Another quantitative measure is the generalized total variance (Delicado, 2001). However, definitions of this type are connected to an underlying stochastic model for the data, which is not used in our case. Therefore we propose a model-independent criterion to assess the quality of a principal curve. We define the *coverage* of a principal curve m by the fraction of all data points which are situated in a certain neighborhood of the principal curve. More precisely, let a principal curve algorithm select a principal curve m consisting of a set P_m of points. Then

$$C_m(\tau) = \#\{x \in X | \exists p \in P_m \text{ with } ||x - p|| \leq \tau\}/n$$

is the coverage of curve m with parameter τ. Obviously the coverage is a monotone increasing function of τ and will reach the value 1 for τ tending to infinity. Note that the coverage can be interpreted as the empirical distribution function of the "residuals", i.e. the shortest distance between data and principal curve. For evaluating the quality of a principal curve fit it is necessary to take a look at the whole coverage curve $C_m(\tau)$. In Fig. 8 we provide the coverage plots for the spiral data (Fig. 4), each for the HS, KKLZ and LPC algorithms and for principal component analysis. For the small spiral, the coverage of the LPC and the polygonal line algorithm from KKLZ are comparable, whereas HS is falling back significantly and is performing only slightly better than the principal component approach. For the big spiral, the LPC algorithm clearly outperforms all other algorithms.

Certainly a concave coverage curve is desirable, i.e. it is “best” when rising rapidly for small τ. The better the principal curve, the smaller is the area in the left top above the coverage curve, i.e. the area between $C_m(\tau)$, the line $\tau = 0$ and the constant function $c(\tau) = 1$. This area corresponds to the mean length of the observed residuals. To obtain a quantitative measure for the performance of a principal curve, we set this area in relation to the corresponding area obtained by standard principal component analysis. The smaller this quotient, the smaller is the relative mean length of the observed residuals and the better is the principal curve compared to principal components. The following table provides this quotient A_C for HS, KKLZ and LPC, where the latter one is calculated applying the optimal bandwidths according to Section 7.

algorithm	small spiral A_C	big spiral A_C
HS	0.79	0.92
KKLZ	0.03	0.66
LPC	0.06	0.08

Table 1: Area-quotient A_C for some principal curve algorithms.

For the HS algorithm, the quotient A_C takes values near 1, which means a quite bad performance. KKLZ yields an excellent value for the small spiral and a rather unsatisfactory value for the big spiral. LPC performs fine in both cases.

7 Selection of parameters

The algorithm is based on two parameters which have to be selected beforehand: The bandwidth h for the radius of the local center of mass and the value t_0 which determines the step length. Assume a center of mass $\mu_{(i)}^x$ at step i, using the data within a radius h around a nearby value $x_{(i)}$. Starting from $\mu_{(i)}^x$, it seems sensitive to walk along the first principal component $\gamma_{(i)}^x$ until the border of the circle around $x_{(i)}$ is reached, what leads

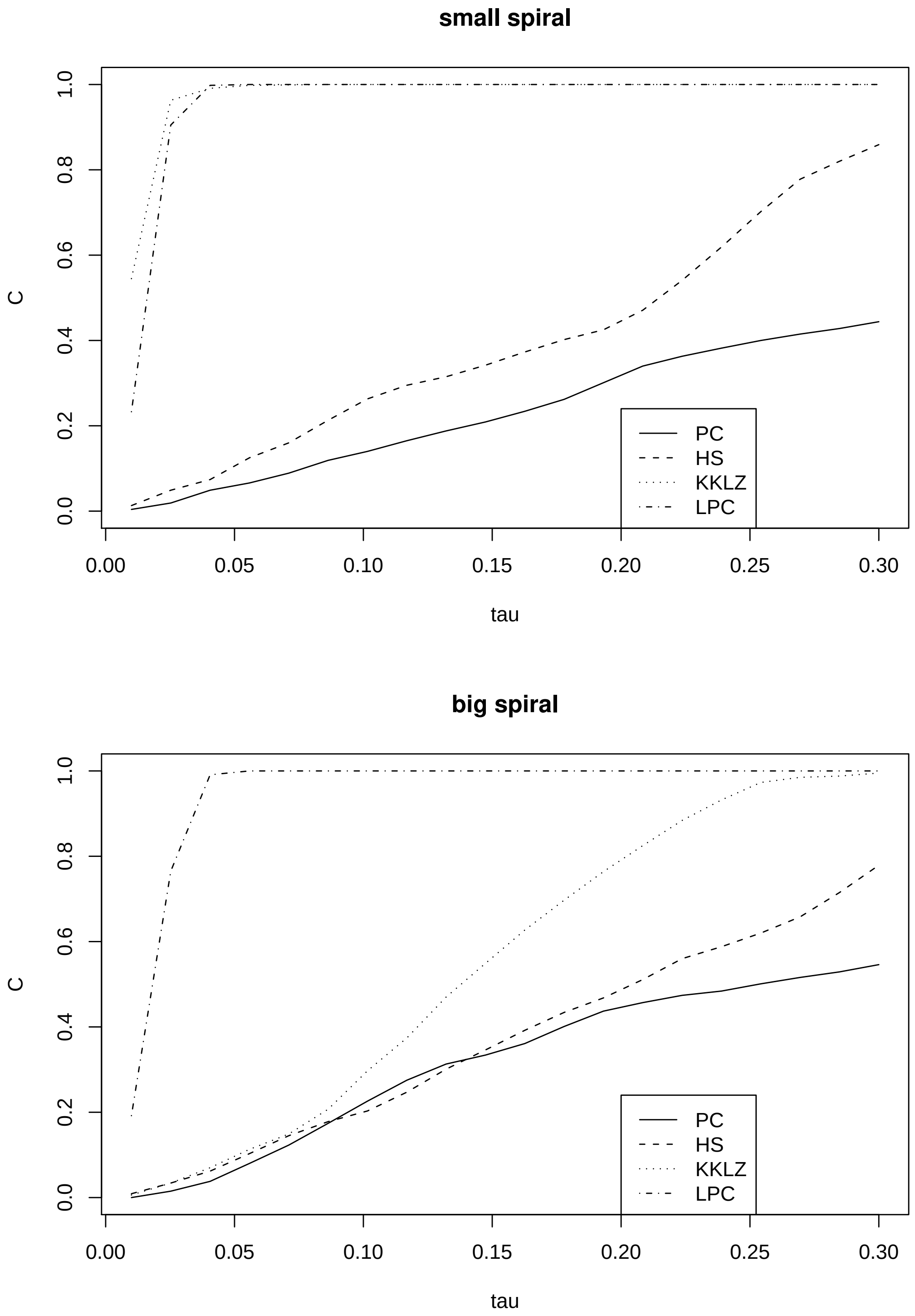


Figure 8: Coverage for small (top) and big (bottom) spiral data.

roughly to the update rule

$$x_{(i+1)} = \mu^x_{(i)} + h\gamma^x_{(i)}.$$

This means that we employ

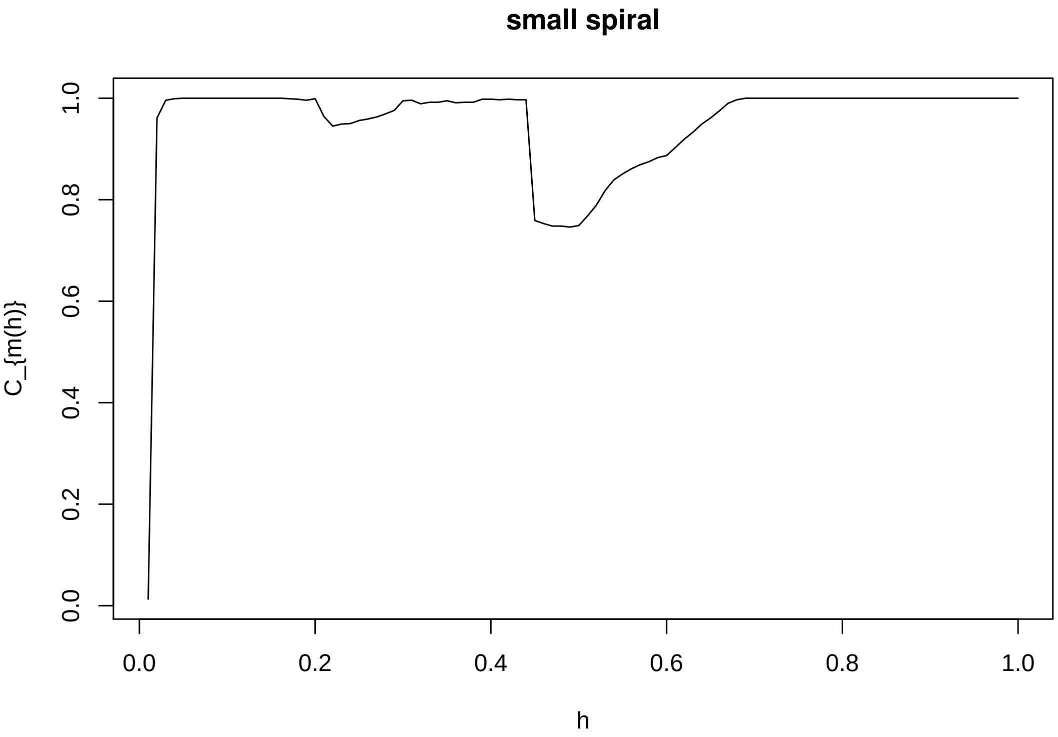
$$t_0 = h.$$

This rule works fine in practice and was applied in all examples in this paper. It now remains to select the value h, which plays the role of a classical smoothing parameter, thus the smaller the value of h, the more details are unveiled by the local principal curve and the more wiggly it is. To select h, we make use of the concept of coverage introduced in the previous section. The idea is the following: If a certain bandwidth h is supposed to serve satisactory for calculating the local center of mass around x, we assume implicitely that this value h covers more or less the width of the data cloud around x. Thus, as a criterion for the adequacy of a principal curve $m(h)$ calculated with a certain bandwidth h we can apply its proper coverage $C_{m(h)}(h)$. We will will refer to this coverage as *self-coverage* hereafter. This curve has a typical behaviour: It starts with small values, then increases rapidly until a local maximum is reached, where the best fit is achieved. Afterwards the self-coverage curve is *falling* again or shows erratic behaviour, but finally rises up to 1 since for large bandwidths the coverage naturally takes the value 1. Note that the fact that $C_{m(h)}(h)$ is falling is not in contradiction with the property of monotoneness mentioned in the previous section: In contrast to $C_{m(h)}(h)$, the coverage $C_m(\tau)$ is calculated for the *same* principal curve m for all τ! Our parameter selection rule is the following:

Choose the *lowest* parameter h for which

- the function $C_{m(h)}(h)$ achieves its first local maximum,
- or, if no local maximum exists, the function $C_{m(h)}(h)$ achieves the value 1.

We want to illustrate this methodology by means of the spiral data shown in Fig. 3. For the small and the big spiral, we calculate the self-coverage over a grid from $h = 0.01$ up to $h = 1.0$ in steps of 0.01. The results are presented in Fig. 9. Since the maxima are partially very flat, we provide in addition the numeric values (for the crucial range of h) in Table 2.



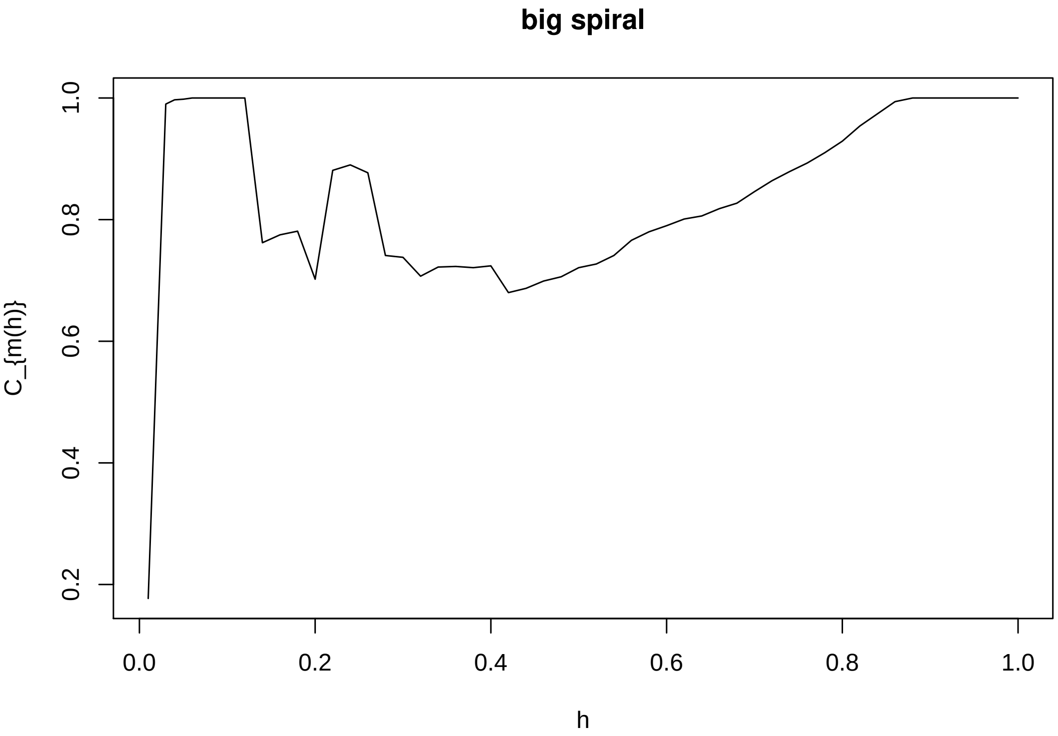


Figure 9: Self-coverage for spiral data.

	small spiral	big spiral
h	$C_{m(h)}(h)$	$C_{m(h)}(h)$
0.01	0.013	0.177
0.02	0.961	0.589
0.03	0.996	0.990
0.04	0.999	0.997
0.05	**1.00**	0.998
0.06	1.00	**1.00**
0.07	1.00	1.00
0.08	1.00	1.00
0.09	1.00	1.00
0.1	1.00	0.998

Table 2: Self-coverage for spiral data.

For the small spiral, the first local maximum is achieved at value $h = 0.05$ with $C_{m(0.05)}(0.05) = 1$. Thus we choose $h = 0.05$. For the whole span from $h = 0.05$ to $h = 0.16$ we would however obtain an ideal principal curve (see the flat maximum). Afterwards the self-coverage is unstable and partially deteriorating. For large values of h the self-coverage tends to 1. The big spiral data lead to a first local maximum starting at $h = 0.06$. Afterwards the curve shows erratic behaviour and approaches slowly to the constant value 1, which is reached at $h = 0.88$. In these calculations, we worked with one fixed starting point (more starting points should not be necessary, since the data cloud is connected and consists of only one branch).

8 Discussion

We demonstrated that the concept of applying local principal components in connection with the mean shift is a simple and useful tool for calculating principal curves, which shows mostly superior performance in simulated data sets compared to other principal curve algorithms. We showed that the algorithm works in simulated and real data sets even for highly complicated data structures. This includes data situations which yet could only be handled unsatisfactory, as data with multiple or disconnected branches. Especially for noisy spatial data as the floodplain data the approach has a high potential to detect the underlying structure. We further provided a tool to select the necessary parameters in a data-adaptive way.

There is still need for further research concerning the theoretical background of the method. Though working fine, we still don't have a theoretical justification why we use *local principal components* to connect the modes of the density. This choice is sensible but in no way unique, and there seem to be many alternatives, such as the extrapolation of the already estimated part of the curve. Due to the nice properties of the mean shift, it might even work to use a line in an *arbitrary* direction, as long it is not orthogonal to the principal curve in the observed point. Important is simply that a movement is made - the mean shift will afterwards adjust the principal curve in direction of the "middle" of the data cloud. However, by applying local principal components the algorithm is fastest, most stable, and the results are as intuitively expected. We consider the first local principal component to be a (biased) approximation of the tangent to the crest line: One can easily derive from its definition that the first local principal component around ξ is the line through ξ which minimizes the weighted distance between the X_i and the line, using the weights k_i as in (4). The first local principal component is therefore that line through ξ that locally gives the best fit.

Furthermore, it will be interesting to investigate if the proposed algorithm can be extended to obtain local prinicpal surfaces or even local principal manifolds of higher dimensions. This might be a quite difficult job, since yet easy techniques as the signum flipping or the mean shift will probably not be transferable to higher dimensional curves without cumbersome extra work.

Acknowledgements

We gratefully acknowledge support from Deutsche Forschungsgemeinschaft (Sonderforschungsbereich 386: Statistical Analysis of Discrete Structures).

References

Aluja-Banet, T. and Nennell-Torrent, R. (1991). Local principal component analysis. *Qüestioó* **3**, 267–278.

Banfield, J. D. and Raftery, A. E. (1992). Ice flow identification in stallite images using mathematical morphology and clustering about principal curves. *J. Amer. Statist. Assoc.* **87**, 7–16.

Buta, R. (1987). The structure and dynamics of ringed galaxies, III: Surface photometry and kinematics of the ringed nonbarred spiral NGC 7531. *The Astrophysical Journal*

Supplement Series **64**, 1–137.

Chang, K. and Ghosh, J. (1998). Principal curves for nonlinear feature extraction and classification. *SPIE Applications of Artificial Neural Networks in Image Processing III* **3307**, 120–129.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intell.* **24**, 603–619.

Comaniciu, D., Ramesh, V., and Meer, P. (2001). The variable bandwidth mean shift and data-driven scale seclection. *8th International Conference on Computer Vision, Vancouver, BC, Canada* **1**, 438–445.

Delicado, P. (2001). Another look at principal curves and surfaces. *Journal of Multivariate Analysis* **77**, 84–116.

Hastie, T. and Stuetzle, W. (1982). Principal curves. *J. Amer. Statist. Assoc.* **84**, 502–516.

Jolliffe, I. T. (1986). *Principal Component Analysis.* New York: Springer.

Kambhatla, N. and Leen, T. K. (1997). Dimension reduction by local PCA. *Neural Computation* **9**, 1493–1516.

Kégl, B. and Krzyzak, A. (2002). Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**, 59–74.

Kégl, B., Krzyzak, A., Linder, T., and Zeger, K. (2000). Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 281–297.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine* **2**, 559–572.

Schölkopf, B. and Smola, A. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10**, 1299–1319.

Skarbek, W. (1996). Local principal components analysis for transform coding. Int. Symposium on Nonlinear Theory and its Applications - NOLTA'96, Proceedings.

Tharpey, T. and Flury, B. (1996). Self-consistency: A fundamental concept in statistics. *Statistical Science* **11**, 229–243.

Tibshirani, R. (1992). Principal curves revisited. *Statistics and Computation* **2**, 183–190.

Verbeek, Vlassis, N., and Kröse, B. (2001). A soft k-segments algorithm for principal curves. *International Conference on Artificial Neural Networks* **??**, ??(to appear).

Wand, M. P. and Jones, M. C. (1993). Comparison of smoothing parametrizations in bivariate kernel density estimation. *J. Amer. Statist. Assoc.* **88**, 520–528.