A k-segments algorithm for finding principal curves

Jakob Verbeek, Nikos Vlassis, Ben Krose

A k-segments algorithm for finding principal curves

J.J. Verbeek, N. Vlassis, B. Kröse *

Computer Science Institute, University of Amsterdam

Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Appeared in Pattern Recognition Letters, volume 23(8), June 2002, pages 1009-1017.

Abstract

We propose an incremental method to find principal curves. Line segments are fitted and connected to form polygonal lines. New segments are inserted until a performance criterion is met. Experimental results illustrate the performance of the method compared to other existing approaches.

Keywords: dimension reduction, feature extraction, polygonal line, principal curve, unsupervised learning

1 Introduction

Principal curves are the nonlinear generalization of principal components. They give a summarization of the data in terms of a 1-d space non-linearly embedded in the data space. Intuitively, a principal curve 'passes through the middle of the (curved) data cloud'. Applications include dimension reduction for feature extraction, visualization and lossy data compression. What concerns the first application, the curve can be used to obtain an ordering of the data by projecting the data to the curve. An example of such a use of principal curves is the ordering of ecological species abundance data [Deáth, 1999]. Applications of the proposed principal curve algortihm can also be envisaged in fitting and recognizing complex shapes in images.

Several definitions of principal curves have been proposed in the literature. One of the earliest definitions is based on 'self-consistency' [Hastie and Stuetzle, 1989], i.e. the curve should coincide at each position with the expected value of the data projecting to that position. Another approach [Kégl *et al.*, 2000] is to define principal curves of length l as curves of length l that achieve the minimum expected squared distance from points to their projection on the curve. The Polygonal

*This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs.

Line Algorithm (PLA) finds principal curves in terms of the latter definition. A more probabilistic approach [Tibshirani, 1992] defines principal curves as curves minimizing a penalized log-likelihood measure. The data is modeled by a mixture of the form:

$$p(\mathbf{x}) = \int_0^l p(\mathbf{x}|t)p(t)\mathrm{d}t \tag{1}$$

where t is a latent variable distributed on an arc-length parameterized curve of length l. $p(\mathbf{x}|t)$ is, for example, a spherical Gaussian modeling the noise located on point t of the curve.

In addition to the above mentioned principal curve algorithms, several other methods can be used to tackle this and closely related problems. The Generative Topographic Mapping (GTM) [Bishop *et al.*, 1998], finds a nonlinear curve embedded in the data space. Along this curve a mixture of Gaussian kernels is placed to generate a distribution on the data space. Also some vector quantization techniques can be used to find approximations to principal curves, e.g. Self Organizing Maps (SOM) [Kohonen, 1995] and Growing Cell Structures (GCS) [Fritzke, 1994]. These methods incorporate a topology among the prototype-vectors. The original space is then mapped to a discrete 1-d space instead of a continuous 1-d space.

Several problems exist with the afore mentioned methods. A common feature of these is that they consist of a combination of 'local models' that are related by a *fixed* topology. When the data is concentrated around a highly curved or self-intersecting curve, these methods exhibit poor performance. This is due to the fixed topology among the local models and to bad initialization. Also, often one does not know a-priori how many 'local models' one needs and one has to make an (educated) guess for the number of local models. On the other hand, the definition of [Hastie and Stuetzle, 1989] explicitly excludes self-intersecting curves as principal curves. In general PLA and Tibshirani's method also fail due to bad initialization in cases where the data is concentrated around highly curved or self-intersecting curves. See Figure 1 for an illustration (for PLA, the dashed curve passing through the data is the generating curve).

In this paper, we use a probabilistic setting to find principal curves by means of maximizing log-likelihood, resembling the method of [Tibshirani, 1992]. We assume the data is corrupted by some noise, therefore we do not require the curve to fit the data exactly. We use the model (1), where $p(t)$ is uniform along the curve and $p(\mathbf{x}|t)$ is a spherical Gaussian located on point t of the curve, with constant variance over all t. The variance is a smoothing parameter, to be set by the user. We restrict the considered curves to polygonal lines (PLs). Our method avoids many of the poor solutions mentioned above (and shown in Figure 1) by fitting local linear models without any topological constraints. To enable an efficient search among PLs with a varying number of segments and to avoid local minima, we use an incremental method that considers PLs consisting of an increasing number of segments. Hence, it starts with models that are relatively easy to fit and gradually increases their complexity.

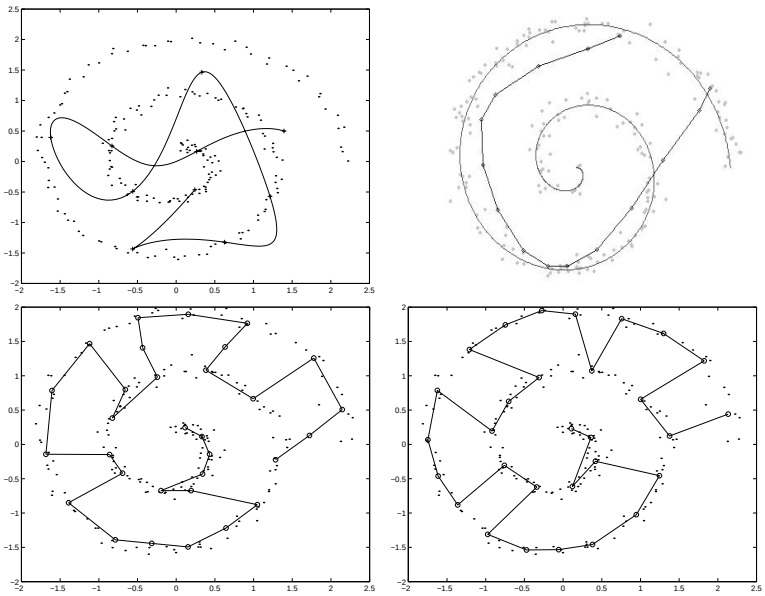

Figure 1: A spiral with some noise. Results for (left to right, top to bottom): GTM, PLA, GCS and SOM.

Overview of the method. By means of local Principal Component Analysis (PCA) we find local linear models. The local linear models are fitted by a (two-step) iterative update scheme. After fitting, the local linear models are combined to form a PL by means of a heuristic search. Next, if appropriate, a new local linear model is added. The location of the new local linear model is found by means of a global search over the data set. Among all PLs encountered by the algorithm, the PL that maximizes an approximation of the log-likelihood of the data is selected.

Overview of the paper. In the next sections we discuss the algorithm in detail. Section 2 is concerned with how we fit the local linear models. In Section 3 we describe the method to find 'good' PLs given a set of line segments. In Section 4 the objective function is considered. We end the paper with a discussion.

2 A k-segments algorithm

In order to find a 'good' PL for the data, we first search for a set of k 'good' line segments. In the first subsection we discuss how we extend the k-means algorithm [Gersho and Gray, 1992] to an algorithm for finding k lines. In Section 2.2 we adapt the k-lines algorithm to find line *segments* that can be used for PL construction. In the last subsection we describe a method to insert a new segment given k segments and a data set.

2.1 Extending k-means to k-lines.

The Generalized Lloyd Algorithm or 'k-means' is a well-known vector quantization method. To extend this method to k-lines, we start with some definitions. A line s is defined as: $s = \{\mathbf{s}(t)|t \in \mathbb{R}\}$, where $\mathbf{s}(t) = \mathbf{c} + \mathbf{u}t$. The distance from a point $\mathbf{x}$ to a line s is defined as:

$$d(\mathbf{x}, s) = \inf_{t \in \mathbb{R}} \|\mathbf{s}(t) - \mathbf{x}\|. \tag{2}$$

Let X_n be a set of n samples from $\mathbb{R}^{\mathrm{d}}$. Define the Voronoi Regions (VRs) [Gersho and Gray, 1992] $V_1, \ldots, V_k$ as:

$$V_i = \{\mathbf{x} \in X_n |\ i = \arg\min_j d(\mathbf{x}, s_j)\}. \tag{3}$$

Hence, V_i contains all data points for which the i^{th} line is the closest, see Figure 2 for an illustration. Analogue to k-means, the goal is to find k lines $s_1, \ldots, s_k$ that minimize the total squared distance of all points to their closest line:

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in V_i} d(\mathbf{x}, s_i)^2. \tag{4}$$

To find lines that are local optima of (4) we modify the k-means algorithm slightly: Start with random orientations and locations of the k lines. Next, alternate between the following two steps until convergence:

1. determine the VRs.
2. replace the lines by the first Principal Component (PC) of their VR.[1]

To prove that this algorithm converges, we make two observations:

- The objective function (4) decreases both in step 1 and step 2. For step 1 this follows from the definition of the VRs. For step 2 this follows from the reconstruction error minimization property of the first PC [Ripley, 1996].
- Due to the finite cardinality of X_n, there are only a finite number of distinct (Voronoi) partitions of X_n.

From these two observations it follows trivially that the algorithm achieves a local optimum for (4) in a finite number of steps.

2.2 From k-lines to k-segments.

Since we want to construct a polygonal line we actually do not search for lines but for line *segments* instead. We therefore replace step 2 of the algorithm. Instead of using the first PC, we

[1] More precisely: we first translate the data in the VR to obtain zero mean and then take the line along the first PC of the translated data.

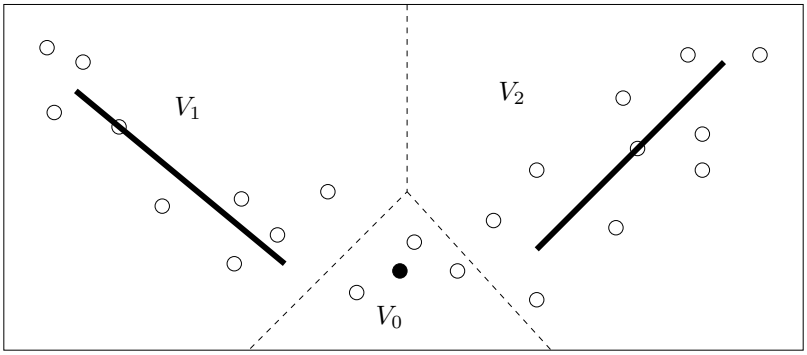

Figure 2: An example where the data (open dots) is partitioned in three Voronoi Regions for two normal segments (V_1 and V_2) and one (V_0) 'zero-length' segment (black dot).

use a segment of it. For example, we could use the shortest segment of the first PC such that all orthogonal projections to the first PC of the points in the VR are included in the segment. We adjust the definition of the VR straightforwardly by using the distances to the segments, i.e. t is restricted to a bounded interval in (2). Using these segments the value of the objective function is not altered. However, two observations suggested another approach:

- The objective function (4) is an easy to compute substitute for the log likelihood of the data given the segments, see Section 4. The length of the segments is not reflected in (4). As a result relatively long segments are found which result in sub-optimal PLs.
- In applications of the algorithm we sometimes observed poor performance. This was due to local optima of (4) which could be avoided by removing the restriction that the segments have to include all projections of the points in the VR to the first PC.

To include explicit optimization of the segment length would be computationally very costly. Therefore, we settled for a heuristic that works well in practice. Experimentally, we found that in general good performance is obtained if we use segments of the first PC that are cut off at $3\sigma/2$ from the centroid of the VR, where σ^2 is the variance along the first PC. To maintain the convergence property, we check whether (4) is actually decreased by the new segment. If (4) is not decreased, we use the segment that includes all projections to the first PC to obtain guaranteed decrease of (4).

2.3 Incremental k-segments.

In most applications we do not know in advance how many segments are 'optimal' to model the data. Therefore, it makes sense to try (many) different numbers of segments and use some criterion to select the best number. Since running the algorithm for many different numbers of segments

is time consuming, we propose to use an incremental strategy. We start with $k = 1$ segment and use the search method described in the previous subsections to find optimal segments. Next, we insert a new segment and optimize the $k + 1$ segments again. The scheme is repeated until some criterion is met (i.e. a predefined maximum of lines is reached or some performance criterion is met).

To determine where to insert the new segment, we first compute for each $\mathbf{x}_i$ in our data set the decrease of (4) if we would place a zero-length segment on $\mathbf{x}_i$. Below we explain why we use this criterion. With a zero-length segment we just mean a point $\mathbf{c}$, hence the distance function (2) reduces in this case to $d(\mathbf{x}, s) = \| \mathbf{x} - \mathbf{c} \|$. The VRs of the zero-length segments are defined as before, by (3), see also Figure 2. To avoid obtaining segments representing too few points, we only consider those zero-length segments for which the cardinality of the corresponding VR is at least three. Let V_{k+1} be the VR of the zero-length segment maximizing the decrease in (4). Then we insert a segment along the first PC of the data in V_{k+1} which is cut off at $3\sigma/2$ on each side of its mean, where σ^2 is the variance of the data in V_{k+1} when projected orthogonally to the first principal component of the data in V_{k+1}. After the insertion of the new segment, again we are ready to do step-wise optimization of the segments. Note that the zero-length segments are not inserted themselves, but only used to select a good position to insert the new segment.

The decrease in (4) due to a insertion of a zero-length segment provides a lower bound on the decrease due to the segment insertion that is actually performed. To see this, let V_{k+1} denote the VR corresponding to the zero-length segment at $\mathbf{x}_i$. It is a well known fact [Gersho and Gray, 1992] that for a given finite set of points $S \subset \mathbb{R}^{\mathrm{d}}$ the mean $\mathbf{m}$ of those points minimizes the squared distance function: $\mathbf{m} = \arg\min_{\mu \in \mathbb{R}^{\mathrm{d}}} \sum_{\mathbf{x} \in S} \|\mathbf{x} - \mu\|^2$, hence: $\sum_{\mathbf{x} \in S} \|\mathbf{x} - \mathbf{m}\|^2 \leq \sum_{\mathbf{x} \in S} \|\mathbf{x} - \mathbf{x}_i\|^2$. Since the new segment s includes $\mathbf{m}$, the total square distance to the segment is at most the total square distance to $\mathbf{m}$ i.e. $\sum_{\mathbf{x} \in S} d(\mathbf{x}, s)^2 \leq \sum_{\mathbf{x} \in S} \|\mathbf{x} - \mathbf{m}\|^2$. Hence, we have the lower bound. Note that the search for the minimizing $\mathbf{x}_i$ can be implemented efficiently as described in Appendix A.

3 Combining segments into a polygonal line

Until now we have introduced an incremental method to find a set of line segments. The outset, however, was to find a PL maximizing the log-likelihood of the data. In this section we discuss how to link the segments together to form a PL Since there are in general many ways to link the segments together, we settle for a greedy strategy. First we generate an initial PL and next we make step-wise improvements on the initial PL.

To achieve a fast algorithm to find the PL we do not consider the data in the construction of the PL. We define a fully connected graph $G = (V, E)$, where the set of vertices V consists of the $2k$ end-points of the k segments. Also, define a set of edges $A \subset E$ which contains all edges that

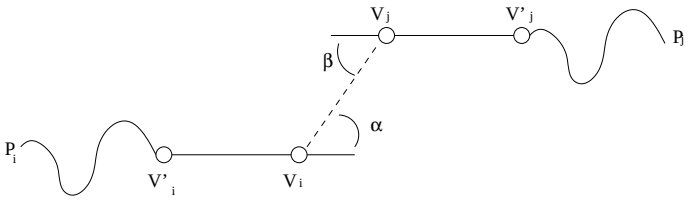

Figure 3: Connecting two sub-paths: the angle penalty of (v_i, v_j) is the sum of the angles between an edge and the adjacent edges, i.e. $\alpha + \beta$.

correspond to the segments. A sequence of edges $\{(v_0, v_1), (v_1, v_2), \ldots, (v_{m-2}, v_{m-1}), (v_{m-1}, v_m)\}$ in which all edges are distinct is called a 'path' [Wilson, 1972]. A path is 'open' if $v_0 \neq v_m$. An open path that passes through every vertex in the graph exactly once is called a 'Hamiltonian path' (HP). Note that we can consider a HP as a set $P \subset E$. We want to find the HP P minimizing the total cost of the path, under the constraint: $A \subset P \subset E$.[2] The cost of a path P is defined as $l(P) + \lambda a(P)$, with $0 \leq \lambda \in \mathbb{R}$ is a parameter to be set by the user. The term $l(P)$ denotes the length of the path, defined as the sum of the lengths of the edges in P. The length of an edge $e = (v_i, v_j)$ is taken as the Euclidean distance between its vertices: $l(e) = \|v_i - v_j\|$. The second term, $a(P)$, is a penalty term equal to the sum of the angles between adjacent edges. The parameter λ controls the trade-off between preferring short paths and paths that do not contain sharp turns. We introduced the $\lambda a(P)$ term to implement a preference for 'smooth' (not having sharp turns) curves. The smaller λ, the smaller the preference for smooth curves. Especially in cases where the generating curve is self-crossing (c.f. Figure 5) setting $\lambda > 0$ is important to find the right PL.

To construct an initial PL we use the greedy strategy outlined below. We call a HP on a subset of V a sub-HP. We start with the original k segments as k sub-HPs. At each step we connect two sub-HPs with an edge e. Note that the total cost of a (sub-) HP consisting of two sub-HPs P_i and P_j linked together by an edge e is the sum of the costs of each sub-HP plus $l(e)$ plus an angle penalty $a(e)$. Figure 3 illustrates the angle penalty $a(e) = \alpha + \beta$ incurred by an edge $e = (v_i, v_j)$ that connects two sub-HPs P_i and P_j. We assign to each edge $e \in (E - A)$ cost $c(e) = l(e) + \lambda a(e)$. The construction algorithm inserts at each step the edge that minimizes $c(e)$ over all edges that connect two sub-paths. Summarizing, the procedure is as follows:

1. Start with k sub-HPs defined by A.
2. While there are at least two sub-HPs
3. Join those two sub-HPs P_i and P_j $(i \neq j)$ by edge $e \in (E - A)$ such that e minimizes $c(e)$ over all edges connecting two distinct sub-HPs.

[2]There always exists a HP for G that contains A since G is fully connected and A connects only pairs of vertices.

Due to the greedy nature of the construction discussed above, it might well happen that we can obtain a HP with lower cost by some simple modifications of the initial HP. To find such improvements we use a simple result from graph theory [Lawler *et al.*, 1985]: finding the optimal (in terms of cost) HP for k vertices, can be expressed as finding the optimal solution for a $k+1$ cities traveling salesman problem (TSP). We use the 2-opt TSP optimization scheme [Lawler *et al.*, 1985] to improve our initial HP.

4 Objective function

Using the incremental method described above, we obtain a sequence of PLs with an increasing number of segments. As stated in the introduction, we search for the PL that maximizes the log-likelihood of the data.

Consider a PL of length l as a continuous arc-length parameterized 1-d latent variable t embedded in $\mathbb{R}^{\mathrm{d}}$, where the embedding is given by $\mathbf{f} : [0, l] \rightarrow \mathbb{R}^{\mathrm{d}}$. For simplicity, we assume a uniform distribution $p(t) = 1/l$ on t here for $t \in [0, l]$. Furthermore, let $p(\mathbf{x}|t)$ be a spherical Gaussian distribution with mean point t on the PL and covariance matrix $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is the $d \times d$ identity matrix. It turns out that, for a latent variable distributing uniformly over a line segment s of length $l \gg \sigma$, the negative log-likelihood for a point $\mathbf{x}$ can be roughly approximated by

$$\log l + d(s, x)^2/(2\sigma^2) + c, \tag{5}$$

where c is some constant dependent on σ. This can be seen as follows:

$$-\log p(\mathbf{x}) = -\log \int_{t\in[0,l]} p(\mathbf{x} \mid t)p(t)\mathrm{d}t = \log(l) + c_1 - \log \int_{t\in[0,l]} \exp(\frac{\parallel \mathbf{x} - \mathbf{f}(t) \parallel^2}{2\sigma^2})\mathrm{d}t. \tag{6}$$

We may write $\parallel \mathbf{x}-\mathbf{f}(t) \parallel^2$ as the sum $d_\perp(\mathbf{f}(t), \mathbf{x})^2 + d_\parallel(\mathbf{f}(t), \mathbf{x})^2$, see Figure 4. Since $d_\perp(\mathbf{f}(t), \mathbf{x}) = d_\perp(s, \mathbf{x})$ is constant for different t, we can write the last term in (6) as:

$$d_\perp(s, \mathbf{x})^2/(2\sigma^2) - \log \int_{t\in[0,l]} \exp(-d_\parallel(\mathbf{f}(t), \mathbf{x})^2)\mathrm{d}t. \tag{7}$$

Let $d_\parallel(s, \mathbf{x}) = \inf_{t\in[0,l]}\{d_\parallel(\mathbf{f}(t), \mathbf{x})\}$, then the last integral can be approximated roughly by $c_2\exp(-d_\parallel(s, \mathbf{x})^2/(2\sigma^2))$. Note that $d(s, \mathbf{x})^2 = d_\parallel(s, \mathbf{x})^2 + d_\perp(s, \mathbf{x})^2$, this leads to equation (5).

We neglect the effect of higher density occurring at one side of the PL at places where different non-parallel segments are connected. Hence, we approximate the total log-likelihood of the data as:

$$n \log l + \sum_{i=1}^{k} \sum_{\mathbf{x}\in V_i} d(s_i, \mathbf{x})^2/(2\sigma^2), \tag{8}$$

where l is the total length of the PL.

In practice, if we take the objective function as a function of k, the number of segments, the global minimum often is the first local minimum of the objective function. So, a simple stopping

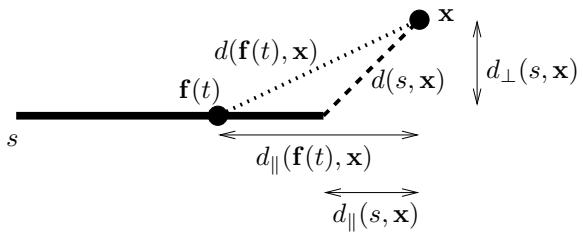

Figure 4: Illustration of the distances between a point $\mathbf{x}$ a segment s and a point $\mathbf{f}(t)$ on s.

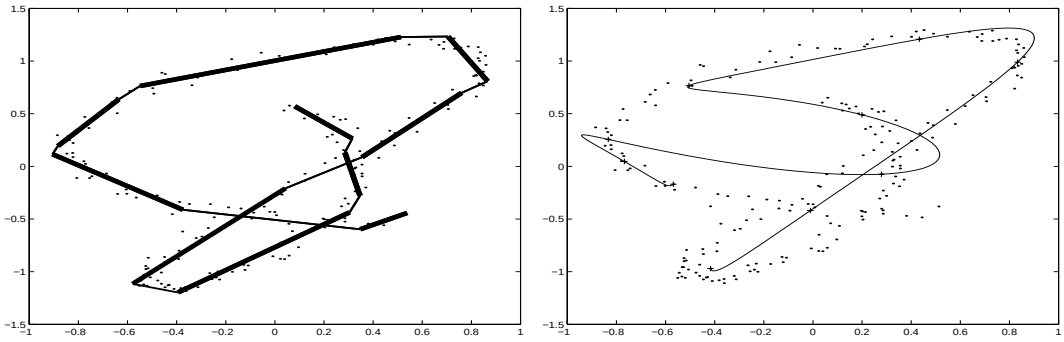

Figure 5: Results on a synthetic data set for our method (left) and GTM (right).

strategy could be to keep inserting segments until the objective function (8) reaches its first minimum. It is also possible to keep inserting segments until some limit $k_{\max}$ on k is reached. A variety of other stopping criteria is possible.

5 Discussion

To test the algorithm presented above, we conducted experiments on several artificial data sets. We used 2-d and 3-d data spaces, crossing curves, non-crossing curves along which we generated data, different amounts of noise and different numbers of training examples. The results are promising, see Figure 5 and 6 for an illustration (the thick segments in the plots are the fitted segments, the thin segments are inserted by the PL construction). Due to limited space we cannot present exhaustive experimental results but just some illustrations here. The good performance is due to (i) the flexibility of the method: the segments are fitted without being connected, and (ii) its incremental nature: segments are added one by one to avoid local optima and (iii) the 'optimal' number of segments is determined automatically. In Figure 6 we illustrate, on the same data as in Figure 1, several stages of the principal curve fitting.

A difference between our method and other principal curve algorithms is the absence of a curvature penalty in our objective function (8). Although a curvature penalty *is* used in the con-

struction of the curves, the algorithm is not too sensitive to changes in its setting and it may even be omitted when modeling non-crossing curves. Regularization of the model complexity is instead introduced in (8) by (i) modeling the distribution of the latent variable by a uniform distribution and (ii) assuming a given level of noise all along the curve. This results in a regularization parameter σ with a clear physical interpretation. The essence is that we avoid (i) distributions on the latent variable that have only finite support[3] and (ii) arbitrarily small noise levels. Experimentally we found that the applicability of the method is not limited to cases where the naive assumption of a uniform distributed latent variable holds. Furthermore, it can be readily replaced by less rigid assumptions on the distribution of the latent variable, for example by a segment-wise uniform distribution on the latent variable.

Note that when one wants to fit other structures than curves, for example closed paths or branching structures, one only needs to replace the search procedure described in Section 3. In the case of closed paths the modifications needed are trivial since the same construction and optimization methods can be used.

Our algorithm has a running time $O(kn^2)$ that is dominated by the insertion procedure, see Appendix A. For each of the k insertions the allocation takes $O(n^2)$ operations and hence takes $O(kn^2)$ in total. Constructing and optimizing all the PLs takes in total $O(k^3)$ (taking the maximal number of iterations of 2-opt fixed). Computing the VRs in the optimization procedure after each segment allocation takes $O(nk^2)$ operations in total. The total number of operations needed by the determination of the principal components is $O(kn)$ (taking the dimensionality of the data space fixed). Since in general $k \ll n$ we arrive at a running time of $O(kn^2)$. In applications involving large amounts of data the quadratic scaling with the amount of data may be problematic. In such cases more sophisticated and faster insertion methods may be used to allow for a $O(k^2 n)$ run time. For example only a small amount of candidate insertion locations may be considered using a method analogue to the one used in [Verbeek *et al.*, 2001].

Some extensions on the presented method suggest themselves:

Smooth curves In some situations there may be a preference for smooth curves over PLs. In such cases one can use the found PL to assign a latent variable value to each datum (implying an ordering on the data). Subsequently, one may use some regression method to find a 'good' smooth curve. The found PL may also serve as an initialization of one of the other methods mentioned that find smooth curves.

Obviate variance estimation The use of an objective function that does not need to be tuned by the user would increase usefulness of the method. We plan to use the Minimum Description

[3]The distribution on the latent variable maximizing likelihood is a distribution with support only at the points on the curve where data projects [Tibshirani, 1992].

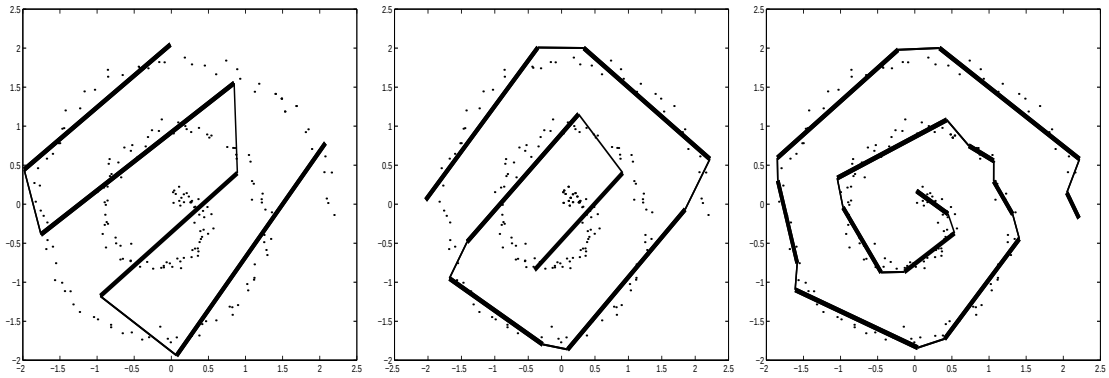

Figure 6: Some of the PLs encountered (using 4, 6 and 12 segments respectively) by the algorithm when fitting data distributed along a spiral.

Length principle to find a parameter-free model selection criterion.

Soft version Currently we are investigating a 'soft' version of the presented method in which the 'hard' Voronoi partitions are replaced by a probabilistic weighting scheme. This way we hope to obtain better performance on very noisy data.

Software implementing the k-segments algorithm in MATLAB can be obtained from the homepage of the first author: http://www.science.uva.nl/ ∼jverbeek.

A Efficient search for allocation of new segment

At the start of the algorithm we compute the pairwise squared distances between all points in the data set, denote the resulting symmetric $n \times n$ matrix by $\mathbf{D}$. In the optimization of the segments, we determine their VRs at each step. To do so we compute distances between all points and all segments, we store for each point $\mathbf{x}_i$ the squared distance d_i^{VR} to the closest segment. Let $\mathbf{D}^{VR} = (d_1^{VR}, d_2^{VR}, \ldots, d_n^{VR})^\top$ and $\mathbf{VD} = [\mathbf{D}^{VR} \ldots \mathbf{D}^{VR}]$ be a square $n \times n$ matrix.

If $\mathbf{A}$ is a matrix and $a \in \mathbb{R}$, then let $\mathbf{M} = \max(\mathbf{A}, a)$ denote the matrix $\mathbf{M}$ where $\mathbf{M}_{i,j} = \max(\mathbf{A}_{i,j}, a)$. Let $\mathbf{G} = \max(\mathbf{VD} - \mathbf{D}, 0)$. Then $\mathbf{G}_{i,j}$ equals the decrease in squared distance for $\mathbf{x}_i$ to its closest segment if we insert the zero-length segment at $\mathbf{x}_j$. Let $\mathbf{1}$ denote the $1 \times n$ matrix $[1\ 1\ \ldots\ 1]$. Then, $\arg\max\{\mathbf{1G}\}$ gives us the index of the datum maximizing the lower bound.

References

[Bishop *et al.*, 1998] C. M. Bishop, M. Svensén, and C. K. I Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[Deáth, 1999] G. Deáth. Principal curves: a new technique for indirect and direct gradient analysis. *Ecology*, 80(7):2237–2253, 1999.

[Fritzke, 1994] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[Gersho and Gray, 1992] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[Hastie and Stuetzle, 1989] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[Kégl *et al.*, 2000] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[Kohonen, 1995] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.

[Lawler *et al.*, 1985] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem*. Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1985.

[Ripley, 1996] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, U.K., 1996.

[Tibshirani, 1992] R. Tibshirani. Principal curves revisited. *Statistics and Computing*, 2:183–190, 1992.

[Verbeek *et al.*, 2001] J.J. Verbeek, N. Vlassis, and B. Kröse. Efficient greedy learning of Gaussian mixtures. Technical Report IAS-UVA-01-04, Computer Science Institute, University of Amsterdam, The Netherlands, May 2001.

[Wilson, 1972] R.J. Wilson. *Introduction to graph theory*. Oliver & Boyd, Edinburgh, 1972.