A soft k-segments algorithm for principal curves

Jakob Verbeek, Nikos Vlassis, Ben Krose

A Soft k-Segments Algorithm for Principal Curves

J.J. Verbeek and N. Vlassis and B. Kröse

Computer Science Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract. We propose a new method to find principal curves for data sets. The method repeats three steps until a stopping criterion is met. In the first step, k (unconnected) line segments are fitted on the data. The second step connects the segments to form a polygonal line, and evaluates the quality of the resulting polygonal line. The third step inserts a new line segment. We compare the performance of our new method with other existing methods to find principal curves.

1 Introduction

Principal curves form the natural generalization of Principal Component Analysis (PCA) [4]. The first principal component can be thought of as the 'optimal' linear 1-d summarization of the data. A principal *curve* generalizes this idea to 1-d *non-linear* summarizations of the data. Since for finite data-sets there are always curves passing exactly through all the data, some regularization of the complexity of the curve is inevitable. Possible applications of principal curves include dimension reduction for feature extraction or visualization of data.

Several definitions of principal curves have been proposed [3, 5]. In [5] the Polygonal Line Algorithm (PLA) is provided as a method to find principal curves. In a probabilistic setting [10] principal curves are defined as curves minimizing a penalized log-likelihood measure. Let $\mathbf{x} \in \mathbb{R}^D$ and assume a mixture density

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

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.

Several other methods from the field of unsupervised learning can also be employed to find principal curves: Generative Topographic Mapping (GTM) [1], Self Organizing Maps (SOM) [6], Growing Cell Structures (GCS) [2] and Isomap [9]. Situations in which the data-set *is* concentrated along a non-linear curve but the aforementioned methods perform poorly or fail completely typically consist of data-sets that are concentrated along a highly curved or self-intersecting curves,

see Figure 1. The poor performance in such situations is mainly due to the local-search strategies employed by these methods.[1] By local-search we mean that the methods start with some initial curve and gradually improve upon it. The result is that the methods end up in rather poor local optima in the space of curves.

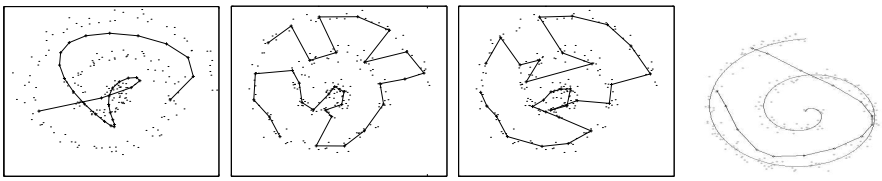

Fig. 1. Results on synthetic data for (left to right) GTM, SOM, GCS and PLA.

We propose an alternative method to find principal curves that is able to overcome some of these local optima. The key feature is that our method can make *jumps* in the space of curves. This is achieved by splitting the search for curves in two parts. First, k unconnected line segments $s_1, \ldots, s_k$ are fitted on the data by means of local-search. Note that we can insert $(k-1)$ line segments to join $s_1, \ldots, s_k$ to form a polygonal line, we can do so in $2^{k-1}k!$ ways. The second search consists of a local search among the $2^{k-1}k!$ candidate polygonal lines. In the next section we discuss exactly what curves we are looking for. The subsequent sections discuss the algorithm in detail. Section 6 ends the paper with a discussion and conclusions.

2 Principal Curves

In this paper, we use a probabilistic setting to find principal curves by means of maximizing log-likelihood, resembling the method by Tibshirani [10]. We use the model (1), where $p(t)$ is uniform along the curve and $p(x|t)$ is a spherical Gaussian located on point t of the curve, with constant variance σ_*^2 over all t. The variance σ_*^2 is a smoothing parameter, to be set by the user. If σ_*^2 tends to zero, the preference of the algorithm is drawn towards curves passing exactly through all data. We restrict the considered curves to Polygonal Lines (PLs).

Suppose our data-set lives in $\mathbb{R}^D$. Let s be a line segment, defined as: $s = \{\mathbf{f}(t) | t \in [-a, a]\}$, with $\mathbf{f}(t) = \mathbf{c} + \mathbf{u}t$, $\mathbf{u}, \mathbf{c} \in \mathbb{R}^D$ and $\|\mathbf{u}\| = 1$. Let $\mathbf{x}_{\|} = \mathbf{c} + ((\mathbf{x} - \mathbf{c})^\top \mathbf{u})\mathbf{u}$ and $\mathbf{x}_\perp = \mathbf{x} - \mathbf{x}_{\|}$. If the PL is a line segment and $p(t)$ and $p(\mathbf{x}|t)$ are as defined above, then (1) can be written as the product: $p_{\|}(\mathbf{x}_{\|})p_\perp(\mathbf{x}_\perp)$, where $p_\perp$ is a (D-1) dimensional spherical Gaussian with zero mean and variance σ_*^2. We approximate $p_{\|}$ with a function that is constant $1/(2a)$ on the interval $[-a+$

[1] This does not hold for Isomap. However, Isomap is not able to find self-crossing curves. This is not surprising since Isomap is based on multi dimensional scaling (MDS).

$\sigma_*, a-\sigma_*]$ and drops to zero outside this interval like a Gaussian with variance σ_*^2, hence $-\log p_\parallel(\mathbf{x}_\parallel) \approx \log 2a + \max(\|\mathbf{x}_\parallel\| - a + \sigma_*, 0)^2/(2\sigma_*^2) + \text{const.}$[2] Combining this approximation with $p_\perp$ and letting $d^2(\mathbf{x}, s') = \max(\|\mathbf{x}_\parallel\| - a + \sigma_*, 0)^2 + \|\mathbf{x}_\perp\|^2$ denote the squared distance between $\mathbf{x}$ and the line segment $s' = \{\mathbf{f}(t) | t \in [-a + \sigma_*, a - \sigma_*]\}$ we have:

$$-\log p(\mathbf{x}) \approx d^2(\mathbf{x}, s')/(2\sigma_*^2) + \log 2a + \text{const} \tag{2}$$

We use (2) to measure the negative log-likelihood of the data given a principal curve. We neglect effects of higher density occurring in bends of the curve.

Note that, using our density model, computing the negative log-likelihood involves measuring the *Euclidean distance between a data point and a line segment*, whereas using a multivariate Gaussian, like in [1, 10, 11], we measure *Mahalanobis distance to the centroid.*

3 Local Search I: Good Unconnected Segments

The task of the first part of the algorithm is to fit k line segments to a data-set. In an accompanying paper [12], we discussed how we can extend the well-known k-means or Generalized Lloyd algorithm to a k-lines or k-segments algorithm. Here, we take another approach where we replace the hard partitioning of the data-set with a 'soft' weighting scheme that determines how much a data point contributes to the configuration of a line segment.

We interpret the fitting of the line segments as a mixture density estimation problem which we solve in an EM style. Each segment defines a mixture component of the form (1). Let π_i be the mixing weight of component i and let D denote the dimensionality of the data space. It is easy to derive the following update equations for σ^2 (recall that σ is assumed constant along the curve) and π_i [12]:

$$\sigma^2_{new} = \frac{\sum_{j=1}^{k} \sum_{\mathbf{x}} p_{old}(j|\mathbf{x}) d(\mathbf{x}, s_j)^2}{nD} \tag{3}$$

$$\pi_{i,new} = \frac{\sum_{\mathbf{x}} p_{old}(i|\mathbf{x})}{n} \tag{4}$$

We do not have such update equations for the parameters that determine the length, location and orientation of the segments. However, given the posterior distribution over the components given a data point, $p(j|x)$, we *can* compute the optimal center and covariance matrix if we were fitting a multivariate Gaussian. We do so and derive, heuristically, parameter updates for the line segments. The center of the segment is taken equal to the center of the Gaussian. The direction is taken along the eigenvector with maximal eigenvalue λ_{max} of the covariance

[2] One can approximate $p_\parallel$ closer with a sigmoid function of $\|x_\parallel\|$. However, the Gaussian approximation allows for simplifications in the computation of the log-likelihood that are not obtained when using the sigmoid.

matrix of the Gaussian. The length of the segment is taken as $3\sqrt{\lambda_{max}}$, this value was experimentally. Using longer segments may yield segments that cover subsequent parts of the curve to cross. Shorter segments describe the data less well. In both cases the construction of the curve (see next section) is made more difficult.

This method is not guaranteed to provide a sequence of pdf's with increasing log-likelihood on the data due to its heuristic nature. Therefore, we only update the parameters of the mixture if this results in an increase in log-likelihood. We stop updating if the increase in log-likelihood drops below a certain threshold.

4 Local Search II: Good Combinations of Segments

To achieve a fast algorithm to find the PL from a set of segments, we do not consider the data in the construction of the PL. The only requirements on the PL are that it is reasonably short and does not contain too sharp angles between adjacent segments. To each PL P we assign a cost $l(P) + \lambda a(P)$, where the first term is the length of the PL and the second is the sum of all angles between adjacent segments of the PL. The value of λ determines the importance of a 'smooth' curve. We construct a first PL in a greedy fashion as follows: First,

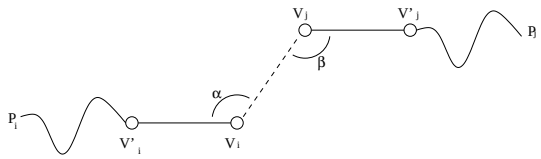

Fig. 2. Connecting two sub-PLs: the angle penalty is the sum of the angles between an edge and the adjacent edges, i.e. $\alpha + \beta$.

we start with all original segments as sub-PLs. Then at each step we connect those two sub-PLs P_i and P_j that can be connected with minimal cost. The cost is given by the length of the connection and λ times the total incurred angle penalty, see Figure 2. We repeat this procedure until just one complete PL is left. The angle penalty is a heuristic used to prevent the construction algorithm from constructing very long curves. Suppose the algorithm can connect a segment s to several other segments at approximately the same distance, then the heuristic articulates a preference for the segment that is most alligned with s. In our experiments we found this heuristic to improve performance significantly, especially when modeling data originating from crossing curves.

Due to the greedy nature of the construction discussed above, it might well happen that we can obtain a PL with lower cost by some simple modifications of the initial PL. To find such improvements we use a simple result from graph theory: finding the optimal (in terms of cost) Hamiltonian Path 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 [7] to improve our initial PL. This procedure is discussed in more detail in [12].

5 Inserting New Segments

When fitting mixture models we often do not know the 'optimal' number of components. Furthermore, the more components we use the greater the risk becomes of getting stuck at a local minimum. Li and Barron [8] recently provided evidence that an approach that interleaves greedy search and component insertion may provide a promising alternative to local search. These reasons motivate why we start our algorithm with one segment and then insert a new segment after both aforementioned local searches (fitting the segments and combining them) have finished. To apply this strategy, we need an efficient method to determine the parameters of a new mixture component given a mixture and a data-set.

Since we cannot compute directly the optimal parameters for the new segment, we use a heuristic to find reasonable initial values for the new segment. We limit the possible locations of the new segment to points in the data-set and choose the location $\mathbf{x}_{i^*}$ that minimizes an upper bound on the total squared distance from data points to the closest segment. We initialize the new component using the data points for which $\mathbf{x}_{i^*}$ is closer than any of the k segments, we denote this subset by X^*. The center of the new segment s_{k+1} is taken as the centroid of X^*. The direction of s_{k+1} is taken along the direction with maximal variance in X^*, denote this variance by λ_{max}. The length of s_{k+1} is taken again $3\sqrt{\lambda_{max}}$. The mixing weight π_{k+1} of the new component is set to $1/(k+1)$, all other mixing weights are rescaled, so that $\sum \pi_i = 1$. The noise variance σ is updated according to $\sigma^2_{new} = (k\sigma^2_{old} + \sigma^2_{k+1})/(k+1)$, where σ^2_{k+1} is the mean squared distance from points in X^* to the new segment.

Alternatively, we could initialize the new component as a spherical Gaussian with variance equal to the variance of the other segments. We could then compute $p(k+1|x)$ and compute new parameters for the new component accordingly.

In addition to computing/guessing initial values for the new component once, we could in an EM style optimize the parameters of the new component and the mixing weight. In our experiments we found that applying this extra optimization does not very much improve and sometimes even worsens performance.

Once the new segment is initialized, we can again start the iterative update scheme discussed in Section 3. The insertion of the new component is not guaranteed to increase the likelihood of the data. Hence, we monitor the change in log-likelihood due to the insertion and only accept the insertion if it leads to increased log-likelihood.

6 Demonstration and Discussion

Experiments To test our algorithm, we conducted several experiments. We compared the performance of our algorithm with the performance of GTM, SOM,

GCS and PLA, see Figure 1. We also compared results with Isomap [9], as expected Isomap is not able to find the crossing structures due to its MDS nature. Data-sets distributed along a number of different curves embedded in both $\mathbb{R}^2$ and $\mathbb{R}^3$ were used. Most of the data-sets used were data-sets on which the aforementioned methods performed poorly. Figure 3 provides results obtained for our method, both on the data used for the other methods (right) and for two other data sets. All other methods failed on the other data sets, we did not include illustrations of these results.

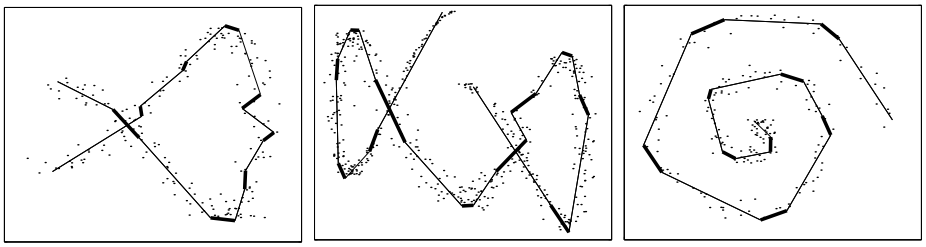

Fig. 3. Some results of our method. In the middle and right-hand picture the thin line segments are the unconnected fitted segments.

Conclusions We observed that our method can avoid poor local minima in many of the situations where the other methods gave poor results. This is due to the combination of local search for the unconnected segments and the local search for good polygonal lines built from those segments. The combination of the two local search steps allows for a non-local search in the space of curves.

The soft weighting scheme used to fit the components as compared to the hard Voronoi partitions used in [12] provides more robustness to the method by making it less sensitive to the details of the configuration of the data-set.

The first local search, the fitting of the line segments, resembles a procedure by Tipping et al. [11]. By using the density model of the form (1) we obtain a mixture density that resembles the final density generated by the curve more closely than when using a simple mixture of Gaussians. We also found

Experimentally we found that if the noise estimation was off up to $1/4$ of the real noise level the algorithm still found correct curves. The weighting factor of the angle penalty is typically set to $\lambda = 1$, although values in the range $(1/2, 2)$ still gave good performance. The 'correct' setting of λ depends on the actual data set.

Note that in cases where one has a preference for smooth curves over the polygonal lines, the algorithm is still of interest. One can simply use the projection indices of the data on the polygonal line as an new variable, and then use standard regression techniques to find a smooth curve for the data.

Future research An objective function that does not require a smoothing parameter σ_* to be set by the user is a subject of further study. We hope to achieve this by implementing an objective function based on Rissanen's Minimum Description Length principle. A different approach to find the 'right' number of segments is to base the objective function on preservation of pairwise distances between (a subset of) the datapoints obtained in a neighbourhood graph.

Finally, we hope to replace some of the heuristic update rules by closed form update equations. Especially for the center and direction of the segments we speculate closed form update equations exist.

Acknowledgment This research is supported by the Technology Foundation STW (project nr. AIF4997) applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

References

1. C. M. Bishop, M. Svensén, and C. K. I Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.
2. B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.
3. T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.
4. I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.
5. 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.
6. T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.
7. 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.
8. J. Q. Li and A. R. Barron. Mixture density estimation. In *Advances in Neural Information Processing Systems 12*. The MIT Press, 2000.
9. J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.
10. R. Tibshirani. Principal curves revisited. *Statistics and Computing*, 2:183–190, 1992.
11. M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.
12. J.J. Verbeek, N. Vlassis, and B. Kröse. A k-segments algorithm to find principal curves. Technical Report IAS-UVA-00-11, Computer Science Institute, University of Amsterdam, 2000.